2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Controlling Internal Structure of Communities on Graph Generator

Hiroto Yamaguchi, Yuya Ogawa, Seiji Maekawa, Yuya Sasaki, Makoto Onizuka
Osaka University, Osaka, Japan
Email: {yamaguchi.hiroto, ogawa.yuya, maekawa.seiji, sasaki, onizuka}@ist.osaka-u.ac.jp

***Abstract*— We propose a novel edge generation procedure, Community-aware Edge Generation (CEG), which controls the internal structure of communities: hub dominance and clustering coefficient. CEG is designed to be adaptable to existing graph generators. We demonstrate the effectiveness of CEG from three aspects. First, we validate that CEG generates graphs with similar internal structures to given real-world graphs. Second, we show how the parameters of CEG control the internal structure of communities. Finally, we show that CEG can generate various types of internal structures of communities by visualizing generated graphs.**

I. Introduction

Graph analysis, such as clustering and classification, has attracted much attention because of their wide adaptation to many applications. In the graph analysis, we need various graphs with community labels for the purpose of evaluating graph analysis techniques. However, the number of such graphs is not large enough for the evaluation. Only few graphs with community labels are available. For example, SNAP provides only eight graph datasets with community labels.

Several graph generators have been recently proposed, such as LFR [4], acMark [6], NetGAN [2], and GraphRNN [8]. LFR and acMark have three functions; (1) community labels are assigned to nodes, (2) various distributions are specified, such as node degree distribution, and (3) graph size is specified arbitrarily. NetGAN and GraphRNN are deep learning-based graph generators that can reproduce realistic structure of given graphs. However, all the above methods do not control various types of the internal structures of communities. The internal structure is characterized by two well-known measurements, hub dominance and clustering coefficient [3], as shown in Fig. 1. To the best of our knowledge, there are no generators that can flexibly control the internal structures of communities since the hub dominance and clustering coefficient are tightly correlated.

We propose a novel edge generation procedure, Community-aware Edge Generation (CEG), which generates various types of the internal structures of communities. Of particular interest, CEG can control clustering coefficient while keeping hub dominance stable by changing the number of *core-nodes*, which we introduce as a parameter. Additionally, we can control hub dominance by changing the number of edges or community size. Also, CEG is designed to be adaptable to existing graph generators. In our demonstration, we show that CEG reproduces the internal structure of real-world graphs more accurately than existing methods. Then, we demonstrate

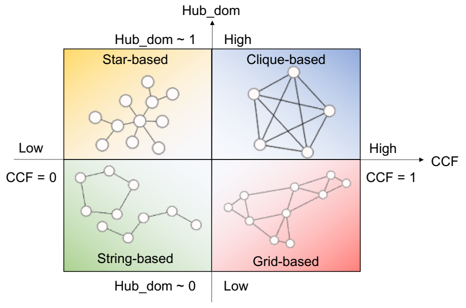


Fig. 1: The types of internal structure of community based on two structural metrics [3].

how the parameters of CEG control the internal structure of communities. Finally, we show that CEG can generate various types of internal structures of communities by visualizing generated graphs.

II. Preliminaries

Let $G = (V, E)$ denotes a graph, where V is a node set, $E \subseteq [V] \times [V]$ is an edge set, $c \in C$ is a community in the graph, and c is composed of a subset of V. We introduce two measurements, hub dominance and clustering coefficient, which characterize the internal structure of communities.

a) Hub dominance: The hub dominance of a community c indicates the ratio of nodes connected with the highest degree node in the community. It is defined as follows.

$$\text{Hub\_dom}(c) = \begin{cases} 1 & (|c| = 1) \\ \frac{\max_{v \in c} d_{int}(v)}{|c|-1} & (otherwise) \end{cases} \tag{1}$$

where $d_{int}(v)$ and $|c|$ denote the number of the internal edges of the node v in the community c and the number of nodes in the community c, respectively.

b) Clustering coefficient: The clustering coefficient of a node quantifies how close its neighbors are to being a clique. We define the clustering coefficient of a single community c, which is defined as follows.

$$\text{CCF}(c) = \frac{1}{|c|} \sum_{v \in c} \frac{2\Delta}{d_{int}(v)(d_{int}(v) - 1)} \tag{2}$$

where Δ denote the number of triangles including node v in the community c. Hereafter, clustering coefficient indicates the clustering coefficient of a community.

c) Characterising internal structure of community: The internal structure of communities can be characterized by the hub dominance and the clustering coefficient, so it is classified into four types, Star-based, Clique-based, String-based, and

Grid-based [3], as shown in Fig. 1. For example, a Clique-based structure has a high hub dominance value and a high clustering coefficient value. That is, a community with Clique-based structure contains many triangles and the highest degree node that connects to most nodes in the community. Other types of the internal structures are also characterized similarly with different hub dominance and clustering coefficient.

d) Difficulty in controlling the internal structure of communities: Existing graph generators can control the global structure of graphs, such as graph size and the distributions of node degree and community size. But they cannot generate various types of internal structures of communities because they uniformly generate internal edges for each community. We could control the hub dominance and clustering coefficient by changing the edge density in communities, which is archived by changing the number of edges and community size. However, the hub dominance and clustering coefficient are tightly correlated each other, so the existing graph generators cannot control them flexibly. For example, when we increase the edge density in a community, both the hub dominance and clustering coefficient increase because the ratio of nodes connected with the highest degree node and the number of triangles increase at the same time. Therefore, we cannot control them flexibly only by changing edge density, since it affects both hub dominance and clustering coefficient.

III. CEG : Community-aware Edge Generation

We propose a novel edge generation procedure, CEG. Our contribution is two-fold. (i) CEG can generate various types of internal structures of communities by controlling the clustering coefficient while keeping the hub dominance stable. (ii) CEG can be easily adapted to existing graph generators, such as LFR [4] and acMark [6]. By adapting CEG to existing graph generators, we can have both benefits of CEG and the existing graph generators: CEG can control the internal structure of communities, whereas the existing graph generators control the global structure, such as the distributions of node degree and community size, and graph size.

A. Approach

We here present how our approach controls the clustering coefficient while keeping the hub dominance stable. The clustering coefficient depends on the number of triangles in the community as defined in Eq (2), so we can control the clustering coefficient by controlling the number of triangles. The communities consist of two types of nodes; *core-node* and *whisker-node* [5]. The core-nodes are high-degree nodes that can connect with any nodes and compose the core of a community when they connect each other, while the whisker-nodes are low-degree nodes that connect to the periphery of the cores. Our idea is based on the fact that many triangles are often formed by connecting the core-nodes. We can control the number of triangles by changing the number of core-nodes. We design CEG as follows. We assign the nodes with the highest degrees in a community to core-nodes and we connect them with any nodes including the whisker-nodes by following the preferential attachment [1], while we assign

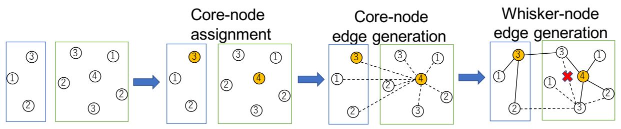


Fig. 2: An overview of CEG

the other nodes to whisker-nodes and we connect them with low-degree nodes, which is inspired by the anti-preferential attachment [7]. Notice that the above approach does not affect the hub dominance, since it does not change the number of internal edges of the highest degree node in communities (See Eq (1)).

Consequently, CEG controls the clustering coefficient by changing the number of core-nodes. Additionally, we can control the hub dominance by changing the number of edges or community size.

B. Algorithm

We present an algorithm of CEG for controlling the internal structure of communities. CEG employs the same idea of anti-preferential attachment [7] but CEG is extended to be community-aware to control the clustering coefficient while keeping the hub dominance stable. Fig. 2 shows an overview of CEG. Our algorithm consists of three steps; core-node assignment, core-node edge generation, and whisker-node edge generation. Since we assume that CEG is adapted to existing methods, community labels and node degrees are generated by the existing methods beforehand. CEG requires two additional parameters: 1) the number of core-nodes h_c for each community c, and 2) the percentile R of low-degree nodes to which whisker-nodes can connect. In the core-node assignment step, the highest degree h_c nodes are assigned to core-nodes for each community c, and the other nodes are assigned to whisker-nodes.

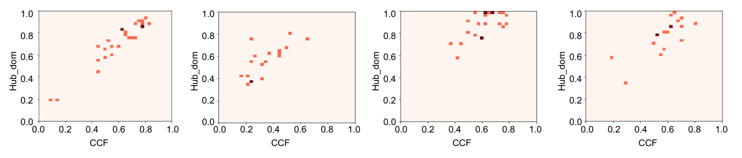
In the core-node edge generation step, CEG connects the core-nodes to the higher-degree nodes (target nodes) in the same community with higher probability, so that the core-nodes compose the core of the community. CEG generates edges for each core-node in descending order of its expected degree so that high-degree nodes likely satisfy their expected degrees.[1] CEG connects each core-node with target nodes that are chosen from all nodes including whisker-nodes based on the following weights:

$$W_i = \theta_*(1 - exp(- < U_i, U_* >)), \tag{3}$$

where i denotes the core-node, θ_* indicates the expected degree of a candidate (*) for target nodes, and U denotes the cluster membership proportions generated in the community assignment step of existing methods, such as acMark.[2] CEG connects nodes in the same community with high probability by adapting the Poisson distribution to the inner product of

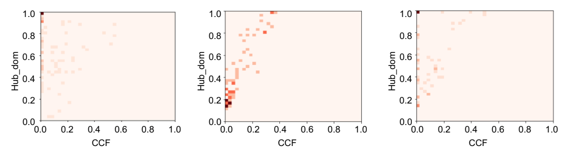
[1]If we generate edges in ascending order of the core-node' degree, it may fail to satisfy its expected degree because there may not be enough target nodes that can be connected to the core-node.

[2]If cluster membership proportions are not generated, such as in LFR, we can assume all nodes belonging to the same community to have the same cluster proportions.



(a) original (b) NetGAN (c) acMark (d) CEG-acMark

Fig. 3: Reproduction of the internal structure of communities in email



(a) original (b) acMark (c) CEG-acMark

Fig. 4: Reproduction of the internal structure of communities in youtube. NetGAN cannot generate graphs within 24 hours.

TABLE I: The earth mover's distance of the internal structure of communities. NA indicates that the execution did not finish in 24 hours.

Datasets	NetGAN	acMark	CEG-acMark
email	0.31	0.26	**0.21**
YouTube	NA	0.25	**0.13**

cluster membership proportion [6] $(1 - exp(- < U_i, U_* >)$. To connect the core-nodes and other nodes based on the preferential attachment, we multiply the weights by the expected degrees (θ_*).

In the whisker-node edge generation step, CEG connects the whisker-nodes to low-degree nodes. CEG generates edges for each whisker-node in descending order of its expected degree, which is similar to the core-node edge generation step. The low-degree nodes are chosen from nodes that do not satisfy the expected degree and whose expected degrees are lower than R percentile (typically 75 percentile). When we set a small value to R, the number of triangles is suppressed. At this time, the low-degree nodes are chosen based on the weights shown in Eq (3).

We note that it is still difficult for the above algorithm to generate a complete star-based community, i.e. the clustering coefficient is zero and the hub dominance is one. To this end, we create a complete star-based community centered on a core-node by setting only one core-node in a community and its degree larger than the community size.

IV. Experiments

In the experiments, we demonstrate that CEG can generate various internal structures of community. We implement CEG on acMark because acMark can control various types of global structures of graphs, such as graph size, node degree distribution, and community size distribution arbitrarily. We call this implementation CEG-acMark.

A. Reproduction of real-world graphs

We show that CEG-acMark can reproduce the internal structure of communities in real-world graphs. We compare CEG-acMark with acMark to evaluate the effectiveness of CEG, and with the state-of-the-art graph generator, NetGAN, to evaluate the reproducibility of the internal structure of communities.[3] NetGAN learns the distribution of biased random walk from a real-world graph with GAN architecture. In this experiment, the evaluation is made on the major communities whose size is larger than 10 in order not to have the noisy effect caused by many small clusters.

We use two real-world graphs; email[4] and YouTube[5]. Email is a graph that models the relationship between senders and receivers in European research institutions. The numbers of nodes, edges, and communities are 1005, 16706, and 42, respectively. YouTube is a graph that models the friendship of the users on YouTube. We assign communities by using a modularity clustering method. In this experiment, we sample 1% of nodes from the original graph to reduce the computation cost. We set the parameters for graph generation as follows: the numbers of nodes, edges, and communities are set at 11348, 15352, and 79, respectively.

Fig. 3 and 4 visualize the internal structure of communities in the graphs generated by NetGAN, acMark, and CEG-acMark, for two datasets, respectively. The points represent the hub dominance in Y-axis and clustering coefficient in X-axis of the communities. The color shows that the darker the point is, the more number of communities there are. In YouTube, the result of NetGAN is missing because it cannot generate graphs within 24 hours. Table I shows the quantitative error computed by the earth mover's distance between the generated communities and the original communities in the two-dimensional space of the hub dominance and clustering coefficient. The results show the average graph generation error of three trials for each method.

In Fig. 3 and Table I, we observe that NetGAN cannot accurately generate the internal structure of real-world graph communities. In Figure 3b, the generated internal structure of communities has smaller hub dominance and clustering coefficient values than the original graph on the whole. This is because NetGAN does not generate graphs by taking the internal community structure into account, although it can control the global characteristics of the graphs. Also, we observe that the graphs generated by CEG-acMark are closer to the original graphs than those generated by acMark in Table I. However, we cannot see a clear difference in the internal structure of communities in Fig. 3 because the hub dominance and the clustering coefficient are closely correlated each other in the email graph.

Fig. 4 and Table I show that CEG-acMark can reproduce the internal structure of communities more similar in YouTube, compared with acMark. We observe that the hub dominance is high but the clustering coefficient is low in YouTube.

[3]The deep learning-based graph generators, such as NetGAN and GraphRNN, do not generate community labels. However, NetGAN generates nodes that correspond to the nodes in the input graph. Therefore, the community structure can be analyzed with the community labels of the input graph, although NetGAN does not generate community labels.

[4]https://snap.stanford.edu/data/email-Eu-core.html

[5]https://snap.stanford.edu/data/com-Youtube.html

TABLE II: Clustering coefficient by the number of core-nodes: "all" indicates the case where all nodes are core-nodes.

# of core-nodes / community	1	2	5	all
CCF	0.01	0.32	0.53	0.81
Hub_dom	0.99	0.97	0.97	0.99

TABLE III: Hub dominance by the number of edges

# of edges	1K	5K	10K	20K
CCF	0.04	0.32	0.61	0.81
Hub_dom	0.15	0.53	0.84	0.99

acMark cannot reproduce such internal structure of communities because it only generates internal structures that the hub dominance and the clustering coefficient are tightly correlated. In contrast, CEG-acMark can control the hub dominance and the clustering coefficient flexibly, so it precisely reproduces the internal structure of communities.

B. Verification of parameter influence

We show how the parameters of CEG-acMark affect the clustering coefficient and hub dominance. First, we demonstrate that CEG-acMark can control clustering coefficient while keeping hub dominance stable by changing the number of core-nodes. Table II shows that the hub dominance and the clustering coefficient vary with the number of core-nodes. We set the number of core-nodes in each community at the same value. These graphs have 1000 nodes, about 20000 edges, and five communities. In addition, we set the parameters so that the node degrees follow a power-law distribution and the community sizes follow a normal distribution (i.e., the community sizes are almost the same). We generate 10 graphs and show the average of the clustering coefficient and the hub dominance among 50 communities. In Table II, we observe that the clustering coefficient increases as the number of core-nodes in the community increases, while the hub dominance is kept stable even though the number of core-nodes increases.

Second, we demonstrate that CEG-acMark can control hub dominance by changing the number of edges. Note that CEG-acMark can also control hub dominance by changing the community size. We omit the latter demonstration because it has the same effect on the community as described in II-(d). Table III shows that the hub dominance and clustering coefficient vary with the number of edges. In these results, we set the number of nodes and the number of communities at 1000 and five, respectively and set all nodes as core-nodes. Other parameters are the same as in Table II. From Table III, we can see that the hub dominance increases as the number of edges increases. This result indicates that CEG-acMark can control the hub dominance by changing the number of edges. Also, the clustering coefficient also increases as the number of edges increases, as shown in Section II.

Consequently, we can generate graphs that have communities with various hub dominance and clustering coefficient by setting appropriate parameters. For example, CEG-acMark can generate graphs with Star-based communities by setting a large number of edges and a small number of core-nodes.

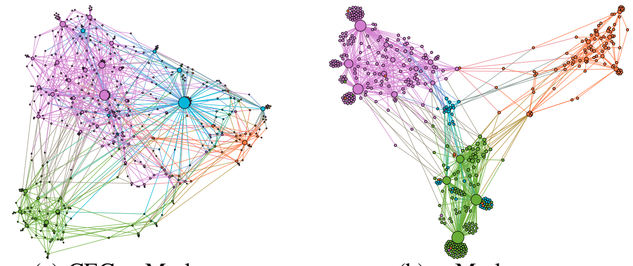

(a) CEG-acMark (b) acMark

Fig. 5: Visualization of generated graphs. The nodes are colored by community labels and their sizes are decided by their degrees.

C. Visualization

To evaluate CEG-acMark can generate various types of internal structures of communities, we visualize graphs generated by CEG-acMark and acMark in Fig. 5.[6] We set the parameters so that the generated graphs have 500 nodes, 1000 edges, and four communities. The nodes are colored by community labels and their sizes are decided by their degrees. In Fig. 5a, the blue community is star-based, and the pink community is close to clique-based since its hub dominance and clustering coefficient are high. The green and orange communities are string-based. These results indicate that CEG-acMark can generate graphs with various types of internal structures of communities. In contrast, acMark generates communities with limited internal structures due to the tight correlation between the hub dominance and clustering coefficient. In Fig. 5b, the blue and orange communities are string-based, and the green and pink communities are close to clique-based.

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers JP20H00583.

References

[1] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *The American Association for the Advancement of Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[2] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in *Proceedings of ICML*, 2018, pp. 610–619.

[3] V.-L. Dao, C. Bothorel, and P. Lenca, "An empirical characterization of community structures in complex networks using a bivariate map of quality metrics," *arXiv preprint arXiv:1806.01386*, 2018.

[4] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *arXiv preprint arXiv:0805.4770v4*, vol. 78, pp. 1–6, 2008.

[5] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proceedings of WWW*, 2008, p. 695–704.

[6] S. Maekawa, J. Zhang, G. Fletcher, and M. Onizuka, "General generator for attributed graphs with community structure," in *proceeding of the ECML/PKDD Graph Embedding and Mining Workshop*, 2019, pp. 1–5.

[7] I. Sendiña-Nadal, M. M. Danziger, Z. Wang, S. Havlin, and S. Boccaletti, "Assortativity and leadership emerge from anti-preferential attachment in heterogeneous networks," *Scientific Reports*, vol. 6, no. 1, p. 21297, 2016.

[8] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proceedings of ICML*, 2018, pp. 5708–5717.

[6]We do not visualize graphs generated by NetGAN because it is specialized for imitating real graphs, so it is not suitable for this evaluation.