Deep Reinforcement Learning for Network Service Recovery in Large-scale Failures

Kazuaki Akashi, Nobukazu Fukuda, Shunsuke Kanai and Kenichi Tayama
NTT Access Network Service Systems Laboratories
NTT Corporation
Tokyo, Japan
{kazuaki.akashi, nobukazu.fukuda, syunsuke.kanai, kenichi.tayama}@ntt.com

***Abstract*—In large-scale failures, telecommunication carriers are required to recover network services as quickly as possible. Since few resources are available in the early stage of the disaster, it is desirable to be able to recover services by repairing as few communication facilities as necessary. This is a combinatorial optimization problem. Recently, methods for solving the network problem using reinforcement learning have been studied. However, existing methods are not sufficient to recover network services because they use only network information for agent learning and decision making. We propose a deep reinforcement learning method that utilizes the services information as well as the network information for minimizing the total repair cost. We represent communication networks and network services as a supply graph and a demand graph, respectively, and use them for agent learning and decision making. Numerical experiments show that the proposed method can reduce the total cost from baselines.**

***Index Terms*—network recovery, large-scale disaster, reinforcement learning, graph neural network**

I. Introduction

Early restoration of network services in the event of a large-scale disaster is an important social issue. When natural disasters such as earthquakes and typhoons occur, various network facilities are damaged over a wide area due to the collapse, flooding, or power outages of buildings in which telecommunications equipment is installed and operated. For example, in the 2011 Great East Japan Earthquake, 385 telecommunication buildings stopped functioning and 90 routes used by relay transmission lines were damaged due to collapses, flooding, and power outages, affecting approximately 1.5 million lines [1]. Telecommunication services have become an important social infrastructure, and are utilized by the national and local governments to respond to disasters and determine the safety of disaster victims. Therefore, telecommunication carriers are required to repair failed communication facilities and recover network services as quickly as possible.

Network service recovery is a combinatorial optimization problem and is discussed as an important topic for fault management. Repairing network facilities incurs costs through using resources such as include workers, vehicles, and communications equipment. In the early stages of a large-scale disaster, the quantity of resources for network restoration is insufficient. Therefore, network services are restored as quickly as possible with minimum cost. This is the problem of finding a combination of failed facilities that minimizes the repair cost to recover the network service.

The basic strategy in conventional researches [5]–[8] is to prioritize repairing facilities on the shortest path for service demands. Figure 1 shows an example of network service recovery. Nodes and links represent communication facilities and cables, respectively. In Figure 1(a), we consider the case of recovering both service demands between nodes 1 and 3 and between nodes 4 and 7. For simplicity, we do not consider the amount of traffic, the capacity of the links, or a routing protocol. Specifically, we assume that each service is recovered when a path connecting the nodes is repaired. Conventional methods based on the basic strategy repairs the nodes on the shortest path of each service demand, as shown in Figure 1(b). However, this strategy cannot repair nodes that are rarely on the shortest path. Therefore, a different approach is needed to obtain a plan to repair the nodes on the detour path as shown in Figure 1(c). Note that (c) is preferable because it has fewer nodes to repair than (b).

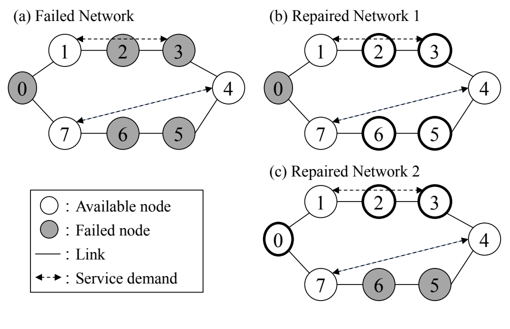

Fig. 1. Example of the network service recovery

In recent years, researchers have solved combinatorial optimization problems in networks using machine learning [2]. Dai et al. proposed solving various optimization problems on graphs by combining graph embedding and deep reinforcement learning (DRL) [3]. As an example of applying a similar approach to network restoration, a method has been proposed for efficiently restoring road networks by combining graph convolutional neural network (GCN) and deep reinforcement learning [4]. However, the application of machine learning has never been proposed for the problem of recovering network services between specific nodes as shown in Figure 1.

We propose a DRL method that uses not only information of the network but also information of services to be recovered. The proposed method represents the services as edges connecting source and destination nodes on the network. Using both the information of the network and the services helps the agent to select appropriate repair nodes.

The contributions of this paper are summarized as follows;

- We introduce a graph representation of network services in order to use service information such as source nodes, destination nodes, and fault conditions.
- We design a novel DRL architecture encoding the network and services information. Using such information, the proposed method minimizes the total cost for recovering network services.
- We conducted experiments with random service demands and node failures and showed that the proposed method outperforms baselines.

This paper is organized as follows. First, we compare related methods and our method in Section II. Second, in Section III, we describe the problem setting and formulation for the network service recovery that minimize the repair cost. Next, in Section IV, we explain the proposed architecture combining graph neural networks (GNNs) and DRL method. Then, we discuss the experimental settings and results of quantitative comparisons between our method and baselines in Section V. Finally, we conclude this paper in Section VI.

II. Related Works

In this section, we review the existing literature on the network recovery problem for large-scale disasters and the reinforcement learning.

A. Network Recovery Problem

Network recovery in the event of a large-scale disaster has been formulated on the minimum-cost network recovery problem and the progressive network recovery problem. In the minimum-cost network recovery problem, we need to find a combination of repair nodes that minimizes the cost required to restore mission-critical services. In the progressive network recovery problem, the repair order is determined to maximize the cumulative traffic until full recovery in the case of repairing failed network in stages. Both of these problems are NP-hard, and existing methods have been proposed to solve them within a practicable time scale.

Bartolini et al. formulated the minimum-cost network recovery problem, and proposed a heuristic algorithm to guide recovery decisions [5]. They introduced a metric of demand based centrality to prioritize each node in their algorithm. The metric generalizes the notion of betweenness centrality [9] which considers connectivity through shortest paths. The method repairs nodes with high centrality, which are considered to greatly contribute to the network services. The demand based centrality is also utilized in the method for solving network recovery problem under incomplete information of failure locations [6], [7]. In addition, Jia et.al proposed a link importance-based network recovery method for large-scale failures in smart grids, which utilized the K-shortest path-based algorithm [10] to calculate the link importance [8]. Although shortest path-based methods are effective, they have a limitation since they cannot share detour paths among service demands.

B. Reinforcement Learning

Dai et al. proposed a Deep Q-Network (DQN) based method for solving various optimization problems on graphs by combining graph embedding and DRL [3]. This method uses a framework that incrementally constructs a solution by iterating graph embeddings and decisions. Specifically, the Q-values of the nodes are computed by a neural network at each iteration, and the one with the largest Q-value is greedily selected.

Fan et al. proposed a method for efficiently recovering road networks [4] with the framework similar to [3]. To solve a road network recovery problem, this method estimates a reward for each action by inputting the network with intersections as nodes and roads connecting intersections as links into a GCN and an artificial neural network. Their method is not directly applicable to recover network services that communicate between specific nodes because it aims to recover traffic among all nodes.

In addition, DRL methods with GNNs have been applied to routing problems [11], [12]. The routing problem is similar to the network service recovery problem in that it connects source (origin) and destination nodes. However, these existing studies do not consider repair of failed nodes. Moreover, these methods transform source and destination information into the betweenness centrality of the links before inputting it into the GNN. Therefore, this method has the same limitations as shortest path-based methods.

In this paper, we focus on node repairing in the minimum-cost network recovery problem. Our proposed method optimizes the recovery plan by directly inputting information about the service demands to the GNN, rather than the importance of the nodes based on the shortest paths between them.

III. Problem

In this section, we describe the network service recovery problem in large-scale failures. Then, we formulate a programming problem that minimizes the total cost to repair all failed services. Here, we assume that the capacity of the links is sufficiently larger than the amount of service demands. Thus, we consider only the connectivity between the source and destination nodes of service demands.

We model the communication network as a supply graph $G_\mathrm{s} = (V, E_\mathrm{s})$, where V and E_s represent nodes and links of the network, respectively. We also model the network services as a demand graph $G_\mathrm{d} = (V, E_\mathrm{d})$, where E_d represent service demands. Each pair $e^\mathrm{d}_{ij} \in E_\mathrm{d}$ has a source node $i \in V$ and destination node $j \in V$. Each node i has a failure status f_i and a repair cost r_i. If a node is failed $f_i = 1$, otherwise $f_i = 0$. Note that, in contrast to the model in [5], a supply graph G_s and a demand graph G_d have the same nodes V.

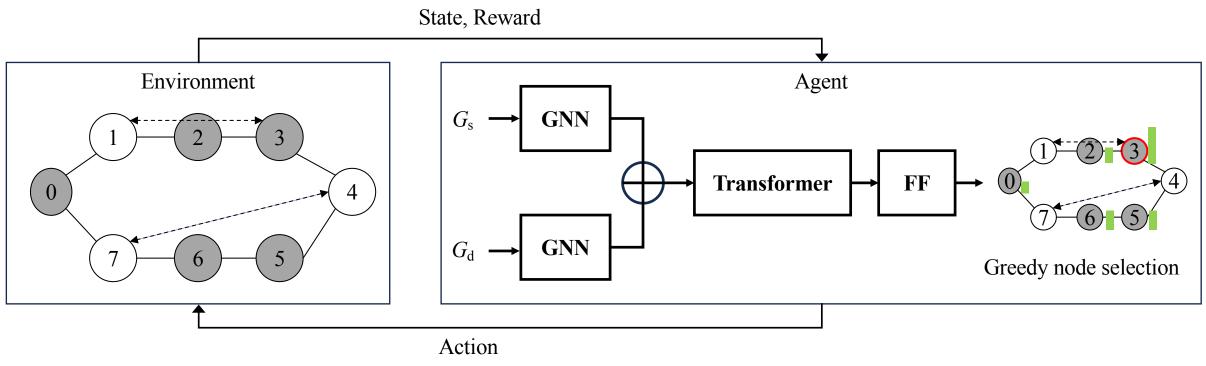

Fig. 2. Overview of SDG-DRL

Specifically, G_d includes nodes without edges. This is to facilitate processing in the DRL.

We define the binary variables δ_i, which represent the decision to repair the node $i \in V$ as (1). Here, $\delta_i = 1$ if node i is repaired, and $\delta_i = 0$ otherwise. Since only the failed node can be repaired, $\delta_i = 0$ if $f_i = 0$ as (2).

$$\delta_i \in \{0, 1\} \;\; (i \in V) \tag{1}$$

$$\delta_i = 0 \;\; (i \in V | f_i = 0) \tag{2}$$

We introduce a constraint (3) which is the connectivities on the network between the source and destination nodes. Here, $\epsilon(i, j)$ is the number of paths between node i and j in the supply graph G_s that pass through only the available nodes. It is a function of binary variables δ_i.

$$\epsilon(i, j) > 0, \; (i, j) \in E_\text{d} \tag{3}$$

The objective in the network service recovery problem (4) is to minimize the total repair cost.

$$\min \sum_{i \in V} r_i \delta_i \tag{4}$$

IV. Proposed Method

We propose a Supply and Demand Graph based DRL (SDG-DRL) method to solve the network service recovery problem for large-scale failures. To solve the network service recovery problem, we use a framework that incrementally constructs a solution, similar to [3] and [4].

Figure 2 is an overview of our proposed method. In the DRL, an agent observes the state of an environment, selects actions, and receives rewards on the basis of its actions. As shown in Fig. 2, we consider the failed network and services as the environment. Moreover, the agent estimates a reward for each action Q(s,a) (green bars) by a model, and selects an action (the highlighted node) on the basis of a greedy policy. More specifically, we define state, action, reward, and termination as follows.

- *State*: The state s is a set of supply graph $G_\text{s}^{(t)}$ and demand graph $G_\text{d}^{(t)}$ at each time step t. Nodes in $G_\text{s}^{(t)}$ and $G_\text{d}^{(t)}$ have attributes such as the node number i, the failure status f_i, and the repair cost r_i. Edges in $G_\text{d}^{(t)}$ have an attribute such as the demand status f_{ij}^d. If the path from node i to j does not exist in G_s excluding the failed node, $f_{ij}^\text{d} = 1$, otherwise $f_{ij}^\text{d} = 0$.
- *Action*: The agent selects and repairs a node. Thus, the action space is the node candidates that can repair at each step. The states of the selected node and the associated service are updated by the action a.
- *Reward*: If the agent's action recover any services, the environment immediately rewards the agent. Specifically, the environment provides greater rewards to agents for actions that recover more services at lower cost. On the other hand, if a service is not recovered, the environment penalizes the agent on the basis of the repair cost.
- *Termination*: Each episode terminates when all the failure services have been restored. The proposed method outputs a set of nodes selected during the episode.

A. Model

We describe the model for estimating the reward of each action using the network and service information.

First, the agent embeds attributes of nodes and service demands. The node embedding $\hat{x}_i^{(t)}$ is calculated using the node number i ,the node status $f_i^{(t)}$, and the repair cost r_i as follows.

$$\hat{x}_i^{(t)} = \text{E}_\text{n}(i) + \text{E}_\text{f}(f_i^{(t)}) + \text{FF}(r_i) \tag{5}$$

where E_n and E_f is an embedding function and FF is a fully connected feed-forward function. Similarly, the service demand embedding $\hat{e}_{ij}^{\text{d}(t)}$ is calculated by using the demand status $f_{ij}^{\text{d}(t)}$.

$$\hat{e}_{ij}^{\text{d}(t)} = \text{E}_\text{e}(f_{ij}^{\text{d}(t)}) \tag{6}$$

Next, the agent obtains topological representations $\hat{h}_i^{(t)}$ GNNs for SDG. The node embedding $\hat{x}_i^{(t)}$ is an attribute of node i in G_s and G_s, and the service demand embedding $\hat{e}_{ij}^{\text{d}(t)}$ is an attribute of edge e_{ij}^d in G_d. We utilize graph attention networks (GAT) [13] as GNNs, which can take into account edge attributes. The representation h'_i of each layer is calculated as Eq.(9) through the edge embeddings Eq.(7) and weights Eq.(8), where a and W are parameters to be trained, and $||$ denotes vector concatenation.

$$e_{ij} = a^{\mathrm{T}}\mathrm{LeakyReLU}(W[h_i||hj]) \tag{7}$$

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) \tag{8}$$

$$h'_i = \sigma\left(\sum_{(i,j)} \alpha_{ij} W h_j\right) \tag{9}$$

In addition, to obtain correlations between nodes that are far apart, the agent encodes the topological representation $\hat{h}_i^{(t)}$ through the Transformer [14]. The Transformer block includes a multi-head attention (MHA) sublayer and a node-wise fully connected feed-forward (FF) sublayer. Each sublayer adds a skip connection and batch normalization (BN). The representation h_i^l of each layer is calculated as Eq.(10), l is a layer that does not share parameters, and $\hat{h}_i$ is an intermediate representation. While the inputs of the first layer are the topological representation $\hat{h}_i^{(t)}$, the outputs of the last layer are the node representations $n_i^{(t)}$.

$$\begin{aligned}\hat{h}_i &= \mathrm{BN}^l(h_i^{l-1} + \mathrm{MHA}(h_1^{l-1}, ..., h_n^{l-1})) \\ h_i^l &= BN^l(\hat{h}_i + \mathrm{FF}^l(\hat{h}_i))\end{aligned} \tag{10}$$

Finally, the agent obtains the value of each action and selects a node greedily. FF block projects the node representations into the action space. Output of this block corresponds to an estimated reward for each action, from which the optimal action leading to the highest reward can be selected. Note that the action to select a node with $f_i = 0$ is masked by adding $-\infty$ to its value.

$$Q(s, a_i) = \mathrm{FF}(n_i^{(t)}) \tag{11}$$

B. Training

To train the model, we use the Double-DQN algorithm [15]. Double-DQN uses an experience replay buffer to store past sequential experiences. At the end of each training episode, the experience is retrieved from the replay buffer and used to update the parameters. We update the parameters of the main network by performing a gradient step to minimize the mean squared loss:

$$\begin{aligned}L &= \frac{1}{B}\left(y - Q(s^{(t)}, a^{(t)}; \theta)\right)^2 \\ y &= r^{(t+1)} + Q(s^{(t+1)}, \underset{a^{(t+1)}}{\mathrm{argmax}}\, Q(s^{(t+1)}, a^{(t+1)}; \theta); \theta^-)\end{aligned}$$

where B is the batch size, $r^{(t+1)}$ is the instant reward of the action $a^{(t)}$, and γ is the discount factor of the future reward. θ and θ^- are parameters of the main networks and target network, respectively. The parameters of the target network are slightly synchronized with the parameters of the main network of each episode in Eq.(12).

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{12}$$

In training, the agent decides on the action for each step in accordance with the ε-greedy policy. Thus, with a probability of ε, a node is selected at random.

In the proposed method, GATs for the supply graph and demand graph had two layers and one layer, respectively. Transformer had six layers, and the MHA had eight heads. The embedding dimension and hidden dimension both were 128. γ = 0.99, τ = 0.005, B = 128, and the learning rate was set to 1×10^{-6}.

V. EVALUATION

In this section, we first discuss the baselines of the network service recovery. Second, we describe the experimental settings. Then, we present the numerical results of the performance of the proposed method.

A. Baselines

BC: This method selects and repairs a node with high betweenness centrality [9] at each step. We calculate the betweenness centrality of a network in Eq.(13).

$$BC_v = \sum_{(i,j)\in E_{\mathrm{d}}} \left(\frac{\epsilon_{\mathrm{s}}(i,j|v)}{\epsilon_{\mathrm{s}}(i,j)}\right) \tag{13}$$

where $\epsilon_{\mathrm{s}}(i,j|v)$ is the number of shortest paths between node i and j. $\epsilon_{\mathrm{s}}(i,j|v)$ is the number of these paths that pass through node v. In this method, the higher the betweenness centrality of a node, the more important it is considered for network service connectivity.

GCN-DQN: This method estimates a reward of each action by a two-layer GCN for the supply graph and a two-layer FF. In [4], the attribute of a node in the supply graph is the number of independent pathways between any network node pairs to solve the road network recovery problem. In this paper, to solve the network service recovery problem, the method utilizes BC_v in Eq.(13) as the attribute of a node. Hyperparameters are set to the same values as in the proposed method.

B. Experimental Settings

To evaluate the proposed method against BC and GCN-DQN, we generate a network and services instances for training and evaluation.

We generate a supply graph G_{s} using well-known Erdős-Rényi, and randomly select pairs of nodes to generate a demand graph G_{d}. We assume that nodes in the G_{s} represent communication buildings and pairs in the G_{d} represent communication buildings which have a point of interface for services that should be restored with priority. We used the same topology of the supply graph in the training and evaluation. In other words, we consider the problem of recovering different services for a fixed network.

In this experiment, the number of nodes was 30, and the probability of generating an edge between each node in supply graph is set to 0.2, and the repair cost of each node is set to 1.0.

First, we generated 1000 training graphs and 1000 validation graphs which set the number of service demands $|E_\mathrm{d}|$ to 10 and the failure probability of a node p to 1.0. These graphs are used to train the SDG-DQN model and the GCN-DQN model. The number of epochs is set to 20, and at the end of each epoch, we evaluate the models in validation graphs and use the parameters of the model with the best performance for evaluation.

Next, we generated 1,000 evaluation graphs for each $|E_\mathrm{d}|$ and p. As mentioned above, the topology of the supply graphs is the same as that of the training graphs. We solved the evaluation graphs with each method and compared the repair costs when all services were recovered.

C. Results

The overall comparison results of the total repair cost are shown in Table I and II. The smallest values are bolded.

TABLE I
TOTAL REPAIR COST ON RANDOM NETWORKS WITH $p = 1.0$.

$\|E_\mathrm{d}\|$	10	15	20	25	30
BC	18.893	23.067	25.5	27.047	28.032
GCN-DQN	26.384	28.247	28.975	29.394	29.585
SDG-DQN	**16.419**	**20.862**	**23.249**	**25.067**	**26.446**

TABLE II
TOTAL REPAIR COST ON RANDOM NETWORKS WITH $|E_\mathrm{d}| = 10$.

p	1.0	0.5	0.2
BC	18.893	9.308	3.597
GCN-DQN	26.384	12.45	4.507
SDG-DQN	**16.419**	**7.675**	**3.033**

Table I shows the average cost for each number of service demands $|E_\mathrm{d}|$ with the failure probability $p = 1.0$. SDG-DQN outperforms all other baselines in all settings. At $|E_\mathrm{d}| = 10$, SDG-DQN improved performance by 13.1% over BC and 37.7% over GCN-DQN. These results indicate that the proposed method can recover services more efficiently than existing methods that utilize the centrality of nodes. And, the SDG-DQN also outperformed the baselines when the number of services $|E_\mathrm{d}|$ was increased from the training. Note that as the number of services increases, the total repair cost approaches 30 because the number of source and destination nodes that must be repaired increases. In addition, the performance of GCN-DQN was lower than that of BC. The betweenness centrality does not include source and destination information, so propagating it by GNN rather obscures important nodes. Here, we can see the benefit of the proposed method considering the network and service information.

Table II shows the average cost for each failure probability with the number of service demands $|E_\mathrm{d}| = 10$. At $p = 0.5$ and $p = 0.2$, the proposed method outperforms the baselines. SDG-DRL uses the parameters of the model trained on the supply graph at $p = 1.0$. This shows that the proposed method can obtain good performance without training for each failure probability. This is because the process of recovering services from a state where all nodes have failed includes a state where some nodes have failed. Note that the training process can be made more efficient by restricting the failure probability and range.

VI. Conclusion

We proposed a deep reinforcement learning method to minimize the cost for recovering network services in the event of large-scale failures. To consider network and service information simultaneously, we represented them as supply and demand graphs, which we used for agent learning and decision making. The model of the proposed method consists of two graph neural networks, a transformer and a feed-forward network. We evaluated the proposed method on a random network under various failure conditions by randomly generating services. The experiments showed that the proposed method can decrease the total cost compared with the baselines.

References

[1] NTT EAST, “Recovering from the Great East Japan Earthquake: NTT East’s Endeavors,” Shinjuku, Tokyo, Japan: NTT EAST, 2012. Online. Available: http://www.ntt-east.co.jp/info/detail/pdf/shinsai_fukkyu_e.pdf, Accessed on: Jun. 2023.

[2] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman, ”Learning combinatorial optimization on graphs: A survey with applications to networking”, *IEEE Access*, vol. 8, pp. 120388-120416, 2020.

[3] H. Dai, E. B Khalil, Y. Zhang, B. Dilkina, and L. Song, ”Learning combinatorial optimization algorithms over graphs,” In *Advances in Neural Information Processing Systems*, 2017.

[4] X. Fan, X. Zhang, X. Wang, and X. Yu, ”A deep reinforcement learning model for resilient road network recovery under earthquake or flooding hazards,” *Journal of Infrastructure Preservation and Resilience*, vol. 4, 2023.

[5] N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri, “On Critical Service Recovery After Massive Network Failures,” *IEEE/ACM Transactions on Networking*, vol. 25, pp. 1-15, Aug. 2017.

[6] D. Zad Tootaghaj, N. Bartolini, H. Khamfroush, and T. La Porta, “On progressive network recovery from massive failures under uncertainty,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 113-126, Mar. 2019.

[7] S. Ciavarella, N. Bartolini, H. Khamfroush, and T. La Porta, “Progressive damage assessment and network recovery after massive failures,” in *Proc. IEEE INFOCOM*, pp. 1-9, 2017.

[8] H. Jia, Y. Gai, D. Xu, Y. Qi, and H. Zheng, ”Link importance-based network recovery for large-scale failures in smart grids,” *Wireless Networks*, vol. 27, pp. 3457–3469, 2021.

[9] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Sociometry*, vol. 40, no. 1, pp. 35-41, Mar. 1977.

[10] D. Eppstein, ”Finding the k Shortest Paths,” *SIAM Journal on Computing*, 28(2), 652-673, 1998.

[11] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio , ”Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case,” In *Computer Communications*, vol. 196, pp. 184-194, 2022.

[12] T. Hara and M. Sasabe, ”Deep Reinforcement Learning with Graph Neural Networks for Capacitated Shortest Path Tour based Service Chaining,” in *Proc. IEEE CNSM*, 2022.

[13] S. Brody, U. Alon, and E. Yahav, “How attentive are graph attention networks?,” In *Proc. ICLR*, 2022.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and Ł. Kaiser, ”Attention is all you need,” In *Advances in Neural Information Processing Systems*, pp. 5998-6008, 2017.

[15] H. v. Hasselt, A. Guez, and D. Silver, ”Deep reinforcement learning with double Q-Learning,” in *Proc. AAAI*, pp.2094-2100, 2016.