2022 IEEE International Conference on
Systems, Man, and Cybernetics (SMC)

GNN-EA: Graph Neural Network with Evolutionary Algorithm

Yun Huang, Chaobo Zhang, Junli Wang*
Key Laboratory of Embedded System and Service Computing (Tongji University),
Ministry of Education, Shanghai 201804, China
{hyun, chaobozhang, junliwang}@tongji.edu.cn

2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-5258-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMC53654.2022.9945338

***Abstract*—Recently, Graph Neural Networks (GNNs) have shown great promise in addressing various tasks with non-Euclidean data. Encouraged by the successful application on discovering convolutional and recurrent neural networks, Neural Architecture Search (NAS) is extended to alleviate the complexity of designing appropriate task-specific GNNs. Unfortunately, existing graph NAS methods are usually susceptible to unscalable depth, redundant computation, constrained search space and some other limitations. In this paper, we present an evolutionary graph neural network architecture search strategy, involving inheritance, crossover and mutation operators based on fine-grained atomic operations. Specifically, we design two novel crossover operators at different granularity levels, GNNCross and LayerCross. Experiments on three different graph learning tasks indicate that the neural architectures generated by our method exhibit comparable performance to the handcrafted and automated baseline GNN models.**

***Index Terms*—graph neural network, neural architecture search, evolutionary algorithm**

I. Introduction

Over the past few years, Graph Neural Networks (GNNs) have become an effective method for leveraging the properties of non-Euclidean data. The key issue in a general GNN framework is the information diffusion mechanism, allowing each node of the graph to update its embedding through the message passing between adjacent nodes. Benefiting from its powerful representational capacity, GNNs have yielded impressive performance in various domains such as object detection [1], recommendation systems [2] and protein interface prediction [3].

However, in most cases, GNN models are sensitive to hyperparameters and specific graph-structured data. For example, the depth of GNNs matters because the representations of nodes within the same connected component may converge to identical equilibriums as the network grows deeper in extreme circumstances, i.e. over-smoothing problem [4], leading to a drastic degradation of performance. Consequently, designing a GNN architecture requires prior domain knowledge and strenuous manual efforts.

It has been demonstrated that Neural Architecture Search (NAS) can automatically generate well-performed neural networks from a pre-defined search space which specifies the potential architectures [5]. Another crucial component of NAS is the search strategy, determining the way to explore the search space. The neuro-evolutionary approaches to designing neural networks can be traced back to [6] and become an alternative to NAS spontaneously. Evolutionary algorithm, as a generic population-based metaheuristic optimization algorithm, typically adheres to the following procedures: firstly initialize several individuals as the primary population, perform genetic operations such as crossover and mutation to maintain the diversity of population, evaluate the individuals through task-specific indicators, and select high-performing individuals.

Despite the successful application on Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), traditional NAS algorithms can hardly design the optimal graph neural architectures, suffering from tremendous computational cost and limitation of structural diversity. GraphNAS [7] is the forerunner to cope with graph neural architecture search, using a reinforcement learning-based method. GraphNAS trains a controller RNN, which generates the description of targeted architectures, to maximize the accuracy of the generated GNN. Instead of constructing search space derived from existing state-of-the-art GNN models, GNAS [8] recently designs Graph Neural Architecture Paradigm (GAP), dividing the search space into smaller granularity: feature filtering operations and neighbor aggregation operations. GNAS realizes the searching process through continuous relaxation of the search space and jointly optimization of the neural architecture as well as its weights.

In this paper, we present Graph Neural Network with Evolutionary Algorithm (GNN-EA), a novel framework for identifying the optimal GNN model automatically. As shown in Fig. 1, the evolution process includes three parts: initialization, training and evaluation, genetic operations . First, we create a population consisting of randomly initialized models, or called as individuals, and train them iteratively to fit the specific task. After each training process, the population is split into two groups, the elite individuals with higher fitness value to be retained and other inferior ones to be removed or mutated. We design two types of crossover operators, GNNCross to ensure the efficiency of structural evolution and LayerCross to complete fine-grained functional evolution. The experiment results show that GNN-EA exhibits comparable performance to the previous state-of-the-art handcrafted and automated GNN models.

Our contributions can be summarized as follows:

- We present an evolutionary graph neural network architecture search strategy based on fine-grained atomic operations. Following the search strategy, we propose GNN-EA, a framework that can achieve adaptive adjustment of neural structures without human intervention.
- We conduct comparison experiments on five real-world datasets to evaluate our method. The results prove that GNN-EA outperforms the previous handcrafted GNN models and shows comparable performance to the state-of-the-art automated GNN models.

* Corresponding author

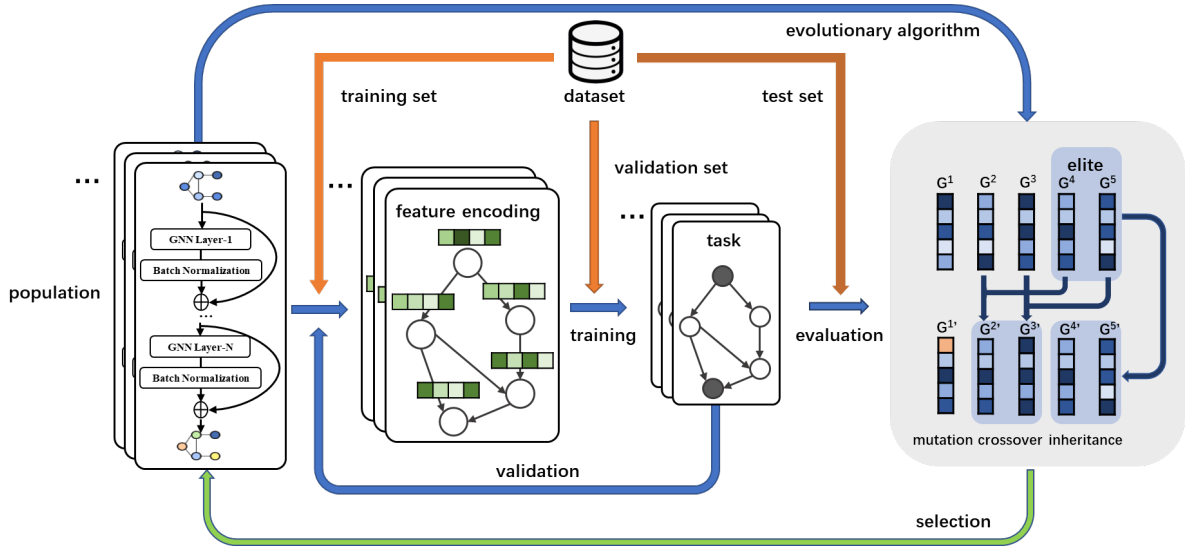


Fig. 1: An overview of GNN-EA training procedure. All individuals in the population are trained on a specific task. According to the performance on the test set, a certain proportion of individuals are selected as elite retaining to the next generation directly and crossing to breed offspring, while the rest are selected to perform the mutation operation. The evolution process is iterative to find the optimal GNN architecture.

II. Related Work

A. Graph Neural Networks

Towards dealing with graph-structured data like social networks, GNNs are initially proposed in [9] as an extension of traditional recursive neural networks. The pioneer GNN models [10], [11] update node representations iteratively with recurrent neural architectures, while convolutional graph neural networks extract information from multi-hop neighbors through stacked convolutional layers. The convolutional graph neural networks can be subdivided into spectral-based [12], [13] and spatial-based [14]–[16] GNNs. GCN [13] utilizes the adjacency matrix of undirected graphs and spectral convolution operations with a non-parametric weight of neighbors to encode local graph structure and node features. To improve extrapolation performance, GraphSAGE [14] proposes an inductive framework that learns functions of sampling and aggregating features from fixed-size neighboring nodes. Inspired by the prosperity of attention mechanism, GAT [15] implements an adaptive aggregation process by performing masked attention on the first-order neighbors. Xu et al. [16] analyze the expressive ability of GNNs theoretically and then propose GIN to distinguish non-isomorphic graphs. To overcome the challenge that previous works often suffer from the limitation of shallow architectures, DeepGCNs [17] introduces residual connections, dense connections and dilated convolutions from CNNs for training very deep GCNs.

B. Neural Architecture Search

With the increasing complexity of neural networks and amount of hyperparameters, Neural Architecture Search , an idea of automating architecture engineering, has received substantial attention recently. The mainstream search strategies can be categorized into Bayesian optimization [18], evolutionary algorithm (EA) [19]–[21], reinforcement learning (RL) [22], [23] and gradient-based [24] methods.

Zoph et al. [22] apply a recurrent network as controller to sample a variable-length descriptive string of neural networks and optimizes the generated "child network" with reinforcement learning. DARTS [24] presents a continuous search space by relaxing the selection of a specific operation to a softmax over all possible operations, so that the searched architecture can be optimized by gradient descent. Real et al. [19] prove the feasibility of employing evolutionary algorithm for optimizing the neural architectures. AmoebaNet-A [21] is the first evolved model to surpass hand-designs on ImageNet by introducing an age property in the evolution process which favors younger genotypes. These above-mentioned methods have significant impacts on respective directions and acquire good performance on discovering promising convolutional and recurrent architectures.

C. Graph Neural Architecture Search

Following policy-based reinforcement learning, GraphNAS [7] first proposes a graph neural architecture search method to design an architecture with independent layers. To avoid the exponential explosion of search space, GraphNAS prunes the combination of atomic operators based on domain knowledge of existing GNN architectures. AGNN [25] incorporates a reinforced conservative controller and a constrained parameter sharing strategy for homogeneous architectures to stabilize the training process. The RNN-based controller increases the computational cost and limits the scalability of search space. Genetic-GNN [26] performs an alternating evolution process involving both neural structures and hyperparameters, represented with a respective string. Focusing on alleviating the over-smoothing problem, AutoGraph [27] is another evolutionary method to automatically generate GNNs, introducing skip connection and layer add to extend the depth. For the same purpose, DeepGNAS [28], based on deep-q-learning algorithm, proposes a two-stage search space including various flexible modules such as residual connection and applies identity mapping in convolutional layers. To tackle the difficulty of handling edge features, AutoGEL [29] explicitly models the link information, improving the results on node and graph classification tasks simultaneously.

TABLE I: Operators of search space

Operators	Values
feature filtering	*dense, sparse, identity, zero*
neighbor aggregation	*max, mean, sum, mlp*

Generally, the construction of search space relies heavily on prior knowledge of well-established networks. Considering that taking components of existing GNNs as atomic operations amounts to a kind of ensemble and fine-tuning, GNAS [8] proposes a fine-grained search space to derive novel network architectures from the original message-passing mechanism. In this paper, we combine the fine-grained atomic operations and the idea of neuroevolution and design an evolutionary search strategy for graph neural architecture search.

III. Methodology

A. Search Space

To reduce the computational overhead and improve searching efficiency, we introduce the Graph Neural Architecture Paradigm proposed in GNAS [8], which defines a tree-topology computation procedure with two types of atomic operations, feature filtering and neighbor aggregation. The neighbor aggregation performs the message passing between nodes and the feature filtering adaptively rescales the message for each node.

In theory, the GAP can simulate most of the existing GNNs approximately and explore architectures with multiple message-passing mechanisms by the flexible combination of the two operations. We adopt the same three-level search space as GNAS to represent the graph neural architecture following GAP, where only the second level is associated with neighbor aggregation operations and all the leaf nodes are incorporated into the output. Table I shows the candidate set of atomic operations. We omit rather extensive formulations of these operations and the topology of computation procedure which have been detailed in [8].

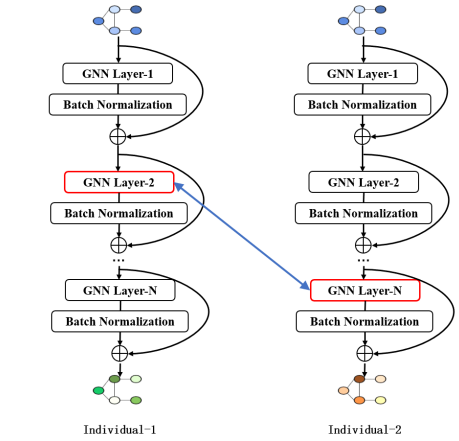
Based on the search space, we stack multiple graph architecture layers as an individual (Fig. 2), with each layer connected through batch normalization and residual connection.

B. Genetic Operators

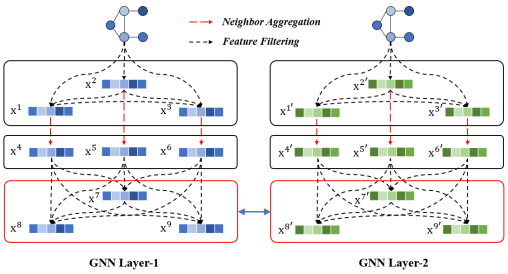
Genetic operators play a vital role in the evolution process to retain excellent genes and ensure genetic diversity. The evolution strategy of GNN-EA consists of three types of genetic operators, namely inheritance, crossover and mutation.

Inheritance. For different individuals, we use the same training data and calculate the fitness value to estimate an individual's ability to propagate its genes. After the training process of each generation, a certain proportion of individuals from the population are selected as elite individuals according to the fitness value, with the rest as inferior ones. These elite individuals directly pass down to the next generation, which preserve their architectures and weights and reduce the retraining cost.

Crossover. Crossover is implemented by exchanging modules between the selected elite individuals. For the tree-topology search architecture, we design two crossover operators at different granularity levels, GNN-level crossover (GNNCross) and layer-wise crossover (LayerCross) to breed offspring. As shown in Fig. 2, GNN-level crossover exchanges



(a) GNNCross



(b) LayerCross

Fig. 2: An illustration of two crossover operators. GNNCross exchanges the graph architecture layer between individuals, while LayerCross exchanges the internal atomic operations between layers.

the whole graph architecture layer of different individuals, while the layer-wise crossover exchanges the internal components of each layer. Essentially, crossover operators aim to merge the excellent properties of parents through the recombination of atomic operations. Particularly, the coarse-grained GNNCross retains and propagates GNN layers of excellent individuals, contributing to searching for desirable general architectures rapidly. On this foundation, the fine-grained LayerCross can recombine and optimize the internal atomic operations of GNN architectures more detailedly.

Mutation. Mutation entails random changes to an individual by altering some modules of its architecture. To ensure the stability of the evolution algorithm, we only select part of inferior individuals to perform the mutation operation and restrict the mutation to the level of atomic operations. That is, in a mutation operation, an atomic operation of each selected individual will be replaced by another randomly picked and initialized one from the candidate set. The mutation operator introduces new atomic operations, bridging the latent gap that some missing atomic operations during initialization cannot be involved in the evolution process.

C. Training Algorithm

Fig. 1 shows the entire training procedure. In GNN-EA, we first randomly select atomic operations to initialize the primary population of individuals according to the tree-topology search architecture. Once the training of current generation finishes, we evaluate the fitness of each individual , adopting accuracy (ACC) for classification task and mean absolute error (MAE) for regression task as evaluating indicators respectively. We

TABLE II: Performance comparison of GNN-EA on five datasets*

Dataset	PATTERN		CLUSTER		MNIST		CIFAR10		ZINC	
Model	#Params	ACC ± STD	#Params	ACC ± STD	#Params	ACC ± STD	#Params	ACC ± STD	#Params	MAE
MLP	0.11M	50.52 ± 0.00	0.11M	20.94 ± 0.00	0.10M	95.34 ± 0.14	0.10M	56.34 ± 0.18	0.10M	0.706
GCN [13]	0.10M	63.88 ± 0.07	0.10M	53.45 ± 2.03	0.10M	90.71 ± 0.22	0.10M	55.71 ± 0.38	0.10M	0.459
GAT [15]	0.11M	75.82 ± 1.82	0.11M	57.73 ± 0.32	0.10M	95.54 ± 0.21	0.10M	64.22 ± 0.46	0.10M	0.475
GraphSAGE [14]	0.10M	50.52 ± 0.00	0.10M	50.45 ± 0.15	0.10M	67.31 ± 0.10	0.10M	65.77 ± 0.31	0.10M	0.468
GraphNAS [7]	0.48M	85.21 ± 0.01	0.48M	52.61 ± 0.22	0.48M	93.80 ± 0.10	0.48M	58.33 ± 0.63	0.48M	0.480
GNAS [8]	0.35M	$\mathbf{86.80 \pm 0.10}$	0.38M	62.21 ± 0.20	0.39M	98.01 ± 0.10	0.41M	70.10 ± 0.44	0.41M	0.276
GNN-EA	0.51M	86.65 ± 0.01	0.53M	$\mathbf{63.03 \pm 0.16}$	0.53M	$\mathbf{99.62 \pm 0.04}$	0.51M	$\mathbf{71.33 \pm 0.08}$	0.51M	**0.218**

* Comparison w.r.t. the number of learnable parameters, the average accuracy on node classification and graph classification task, and the MAE (lower is better) on graph regression task. The results of baseline models depends on the publication data from [30] and [8].

set an elite rate to retain well-performed individuals for inheritance and eliminate low-fitness ones. The new generation is composed of elite individuals and new individuals created by crossover and mutation operators. We iterate this process to maximize the fitness on a specific task and evolve desirable graph neural networks.

The pseudocode description is as the following Algorithm 1, where GEN denotes the number of generations for evolution, N denotes the population size. Specifically, lines 4-8 are the training of each individual, lines 9-10 are the process of evaluation by calculating the fitness, and lines 12-14 are the inheritance, crossover and mutation operations respectively. Then a new generation of population is updated in line 15 and the optimal GNN architecture is outputted in line 17.

Algorithm 1 Training procedure of GNN-EA

Input: training set S, test set S', set of atomic operations M
Output: The optimal GNN architecture G^O

1: Initialize a random population $P = \{G^1, G^2, \ldots, G^N\}$
2: **for** $generation \leftarrow 1$ to GEN **do**
3: **for** $n \leftarrow 1$ to N **do**
4: **for** $epoch \leftarrow 1$ to $epochs$ **do**
5: Get training samples X and labels Y from S
6: Feed data into the model and calculate the loss $loss = Loss(G^n(X), Y)$
7: Update the parameters θ to minimize the loss $\theta \leftarrow Adam(lr, \theta, weight\_decay)$
8: **end for**
9: Get test samples X_{test} and labels Y_{test} from S'
10: Evaluate the fitness as per the performance on S' $fit^n \leftarrow calculate\_fitness(G^n(X_{test}), Y_{test})$
11: **end for**
12: Select the two fittest individuals $G^{child(1)}, G^{child(2)}$
13: Crossover to generate $N-4$ new GNN individuals $G^{child(3)}, \ldots, G^{child(N-2)} \leftarrow GNNCross(G^{child(1)} \times G^{child(2)})$ $LayerCross(G^{child(1)} \times G^{child(2)})$
14: Mutate two of inferior individuals with the lowest fitness value to generate $G^{child(N-1)}, G^{child(N)}$
15: Update the new population $P = \{G^{child(1)}, G^{child(2)}, \ldots, G^{child(n)}\}$
16: **end for**
17: Output the GNN architecture with the highest fitness value

IV. Experiments

A. Datasets and Baselines

Datasets. Following [30] and [8], we conduct experiments on these datasets: ZINC [31] for graph regression task, PATTERN [30] and CLUSTER [30] for node classification task, MNIST [32] and CIFAR10 [33] for graph classification task. ZINC [31] is a real-world molecular dataset focusing on the prediction of constrained solubility, an important chemical property for molecular graph generation. PATTERN [30] aims at recognizing predetermined subgraphs and CLUSTER [30] aims at identifying community clusters, both of which are generated with Stochastic Block Models [34] to model communities in social networks. MNIST [32] and CIFAR10 [33] are popular image classification datasets where images can be converted to graphs using super-pixels [35] and considering the intensity in images as each node's features. More details about the datasets can be found in [30].

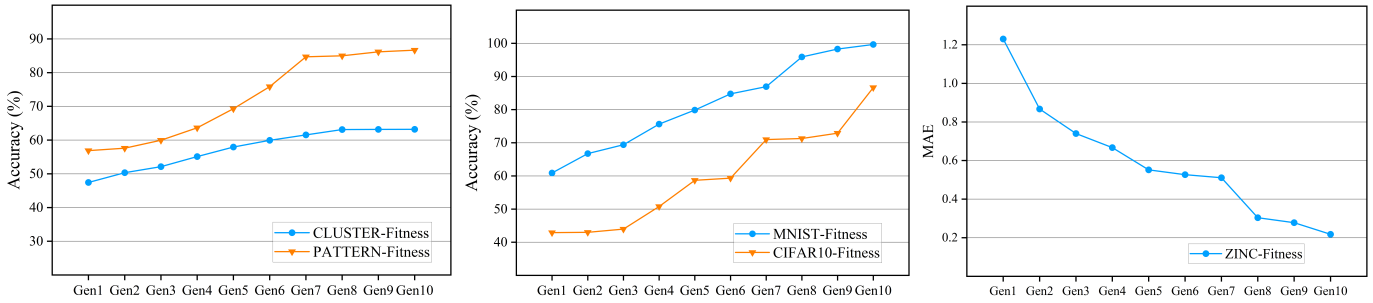
Baselines. For handcrafted GNN models, we select GCN [13], GraphSAGE [14] and GAT [15] as our baselines. GCN [13] and GAT [15] are existing reasonable-designed models with satisfactory results on multiple tasks. GraphSAGE [14] is capable of encoding unseen nodes by sampling and aggregating information from neighboring nodes. Apart from preset GNN models, MLP is another baseline method. Additionally, to verify the effectiveness of our evolution-based method, we also compare with two graph neural architecture search methods, GraphNAS [7] and GNAS [8]. GraphNAS [7] uses reinforcement learning to optimize GNN architectures with an extra RNN controller while GNAS [8] introduces a novel-designed search space and adopts a gradient-based search strategy.

B. Experimental Setting

We set the number of evolution generations as 10, the population size as 8, and the elite rate as 0.25. During each evolutionary step, the elite parents generate two offspring through each crossover operator. We stack 4 graph architecture layers as an individual and each layer is constructed from a three-level search space. To facilitate comparison, the depth of all the baseline models is also fixed to 4 layers. The number of nodes at each level is set as 3 and the hidden dimension is set as 70. Each GNN model is trained for 150 epochs on graph classification and regression task, for 200 epochs on node classification task, with batch size 128. We use Adam with learning rate $lr = 1 \times 10^{-3}$ and weight decay 3×10^{-4} as the optimizer to update parameters.

C. Results

Table II shows the results of comparison experiment. Because of the exorbitant cost of manually designing a GNN model with optimal architecture, the traditional GNNs usually utilize a single message-passing mechanism and set up limited number of hyperparameters. Therefore, GNN-EA shows



(a) Fitness on CLUSTER and PATTERN (b) Fitness on MNIST and CIFAR10 (c) Fitness on ZINC

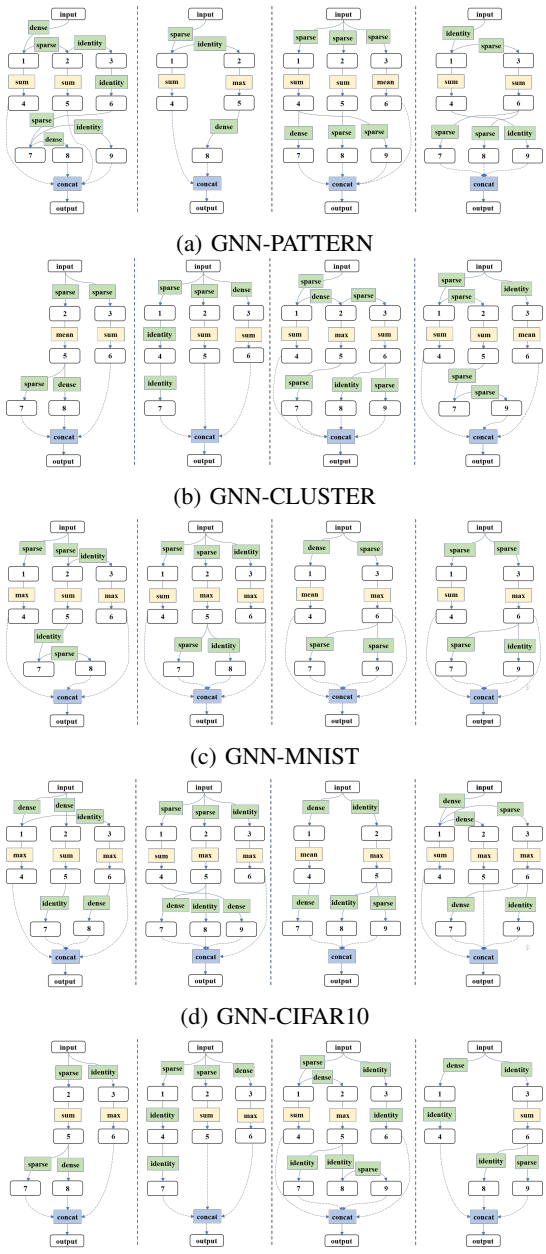
Fig. 3: The variation curve of the best fitness value in evolution process. The horizontal axis represents the generation, while the vertical axis represents the accuracy (%) for node classification (Fig. 3a) and graph classification (Fig. 3b), and the MAE for graph regression (Fig. 3c). Higher accuracy or lower MAE means better performance.

great performance superiority on all three tasks compared to these models. Among NAS-based methods, the performance of architectures searched by GraphNAS varies widely across different datasets. GraphNAS derives atomic operations from existing GNN architectures such as GCN and GAT, where modules with overlapping functions could lead to the latent problem of feature redundancy. Benefiting from the fine-grained atomic operations, GNAS and GNN-EA achieve better results and scalability. The evolutionary search strategy starts with a certain number of moodels, which reduces the impact of suboptimal architectures and the randomness of initialization to some extent. The evolution of population with genetic operators helps to explore the search space more deeply, so that the architectures searched by GNN-EA in our experiments obtain the best accuracy or MAE on nearly all the experiment datasets. Moreover, these models gain smaller standard deviations, which means better stability and robustness. Specifically, the classification accuracy of GNN-EA models can approximately reach 1 on MNIST with decreasing standard deviation at the same time. On graph regression ZINC dataset, GNN models reduce the mean absolute error by 0.488 and 0.262 at most compared to handcrafted GNNs and previous graph NAS methods respectively.

Fig. 3 demonstrates that accompanying with the iteration of evolution steps, elite individuals adapt to the targeted task gradually. The accuracy on CLUSTER and PATTERN levels off after the 8-th generation, indicating the node representations of the two datasets are relatively simple. Overall, GNN models on all datasets achieve optimal performance at the 10-th generation. Due to the limitation of computation resources, we restrict the evolution process to 10 generations. If the hardware permits, GNN-EA is expected to gain further improvement on CIFAR10 and ZINC.

D. Visualization Analysis

The corresponding GNN architectures of each dataset are detailed in Fig. 4. According to the analysis in [8], [16], mean aggregator performs well on graphs where the distributional information is more important than the exact structure, sum aggregator is conducive to capturing structural information, while max aggregator pays more attention to representative nodes. GNN-PATTERN and GNN-CLUSTER focus more on sum aggregator because node classification task requires to integrate and distinguish more information from neighboring nodes. It can be proved by GAT that attention mech-



(a) GNN-PATTERN

(b) GNN-CLUSTER

(c) GNN-MNIST

(d) GNN-CIFAR10

(e) GNN-ZINC

Fig. 4: The optimal graph neural network architectures searched by GNN-EA. Each task-specific model consists of 4 graph architecture layers (from left to right), connected through batch normalization and residual connection.

anisms improve classification accuracy. GNN-MNIST and GNN-CIFAR10 tend to select max aggregator, because for graph classification task, crucial information from some nodes generally plays a decisive role in representing the graph or subgraph rather than those non-essential nodes. This can explain why graph-agnostic MLP gains comparable performance with traditional GNNs. On graph regression task, GNN-ZINC combines the sum and max aggregator because both functional groups and specific atoms control chemical properties. Thus, GNN-EA can explore the combination of atomic operators adaptively for data with different features.

V. Conclusion

In this paper, we aim to study graph neural architecture search based on the idea of neuro-evolution. We propose GNN-EA, a novel graph NAS framework with an evolutionary search strategy designed for exploring more fine-grained search space. According to experiment results, architectures generated by GNN-EA show great competence in searching for optimal GNN models and gain comparable performance to the previous state-of-the-art methods. Further visualization analysis demonstrates the reasonability and effectiveness of our method. Due to the limitation of experimental condition, we don't evaluate our method with more evolution generations and bigger population size. Future work will focus on two aspects, tapping the potential of GNN-EA in a large evolution environment and reducing the searching and training cost. Besides, introducing novel atomic operations to enrich search space is another promising approach to improvement.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (Grant No. 2017YFA0700602).

References

[1] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711–1719.

[2] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 936–10 945.

[3] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6533–6542.

[4] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018, pp. 3538–3545.

[5] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[6] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *ICGA*, vol. 89, 1989, pp. 379–384.

[7] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search." in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, vol. 20, 2020, pp. 1403–1409.

[8] S. Cai, L. Li, J. Deng, B. Zhang, Z.-J. Zha, L. Su, and Q. Huang, "Rethinking graph neural architecture search from message-passing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6657–6666.

[9] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734.

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[11] C. Gallicchio and A. Micheli, "Graph echo state networks," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR*, 2017.

[14] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR*, 2018.

[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR*, 2019.

[17] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9267–9276.

[18] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing, "Neural architecture search with bayesian optimisation and optimal transport," 2018, pp. 2020–2029.

[19] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2902–2911.

[20] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *6th International Conference on Learning Representations, ICLR*, 2018.

[21] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[22] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR*, 2017.

[23] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *5th International Conference on Learning Representations, ICLR*, 2017.

[24] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *7th International Conference on Learning Representations, ICLR*, 2019.

[25] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-gnn: Neural architecture search of graph neural networks," *arXiv preprint arXiv:1909.03184*, 2019.

[26] M. Shi, D. A. Wilson, X. Zhu, Y. Huang, Y. Zhuang, J. Liu, and Y. Tang, "Evolutionary architecture search for graph neural networks," *arXiv preprint arXiv:2009.10199*, 2020.

[27] Y. Li and I. King, "Autograph: Automated graph neural network," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 189–201.

[28] G. Feng, C. Wang, and H. Wang, "Search for deep graph neural networks," *arXiv preprint arXiv:2109.10047*, 2021.

[29] Z. Wang, S. Di, and L. Chen, "Autogel: An automated graph neural network with explicit link information," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[30] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[31] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "Zinc: a free tool to discover chemistry for biology," *Journal of chemical information and modeling*, vol. 52, no. 7, pp. 1757–1768, 2012.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[34] E. Abbe, "Community detection and stochastic block models: recent developments," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6446–6531, 2017.

[35] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.