Multi-objective evolutionary algorithm based graph neural network architecture search *

Lianyi He[1,2,3], Xiaobo Liu[1,2,3,4] **, Hongbo Xiang[1,2,3], and Guangjun Wang[1,2,3]

[1] School of Automation, China University of Geosciences, Wuhan 430074
[2] Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan 430074
[3] Engineering Research Center of Intelligent Technology for Geo-Exploration, Ministry of Education, Wuhan 430074
[4] Key Laboratory of Geological Survey and Evaluation of Ministry of Education, China University of Geosciences, Wuhan 430074
{helianyi,xbliu,hbxiang,gjwang}@cug.edu.cn

Abstract. Graph Neural Networks (GNN) has become a powerful graph data processing method, which has been widely used in node classification, link prediction, and other graph analysis tasks. Due to the diversity and complexity of graph structures and information propagation, as well as the handling of heterogeneous graphs, etc., the design of GNNs presents many challenges. The existing methods have high classification accuracy, but their structures are very complex. In this paper, a graph neural network architecture search framework named MO-GNN, which is based on multi-objective evolutionary algorithm is proposed. Furthermore, a weight sharing strategy MO-GNN-WS is proposed to reduce the resource consumption caused by weight training of different architectures. To verify the performance of the proposed algorithm, experiments on four popular graph datasets are used for transduction and induction tasks. The experimental results show that the MO-GNN-WS algorithm outperforms the most advanced neural network architecture search methods in terms of classification accuracy and resource consumption.

Keywords: Graph neural network architecture search, Multi-objective optimization, Weight sharing.

1 Introduction

Graph Neural Networks (GNN) is a deep learning method for graph structure data classification [1]. In recent years, many graph neural network models

* Supported by National Natural Science Foundation of China (61973285, 62076226), Hubei Provincial Natural Science Foundation of China (Grant No.2022CFB438), and Opening Fund of Key Laboratory of Geological Survey and Evaluation of Ministry of Education (GLAB2023ZR08).

** Corresponding author: xbliu@cug.edu.cn

with different neighborhood aggregation schemes have been developed, including graph convolutional networks [2], graph isomorphic networks [3], graph attention networks [4], and local extreme graph neural networks [5], which have achieved good performance in tasks such as semantic segmentation [6], node classification [7], and recommendation system [8]. Despite the great success of graph neural networks, the design of graph neural networks is a knowledge-intensive and labor-intensive process. Moreover, unlike most CNN models in computer vision tasks, GNNs often show a poor transferability toward different scenarios, and require additional manual adjustments when dealing with new tasks [9].

In order to overcome the challenge of designing neural network models by hand, neural network architecture search has become a hot topic. Motivated by the successful application of neural network architecture search in the field of computer vision, there is a growing interest in searching for a robust and well-performing graph neural network model from a predefined search space. For example, GraphNAS [10] is the first to try to apply the neural network architecture search method to the architecture design problem of graph neural networks. Subsequently, Zhao et al. constructed a novel search space and proposed a framework based on differentiable search, which has certain advantages in accuracy and training efficiency compared with some state-of-the-art deep learning methods [11]. At the same time, by using the pseudo-inverse of the matrix to adjust the number of nodes and the linearization length constant of the virtual equivalent dynamic linearization model, also provides a good scheme to solve the neural network architecture search problem [12]. Genetic-GNN algorithm [13] synchronously optimizes the structure and hyper-parameters, whose performance of the searched model can be comparable to reinforcement learning. These automatically designed GNN architectures have achieved competitive or better performances compared with manually designed GNNs for a single task [15].

The existing graph neural network architecture search algorithms have achieved good results in classification accuracy. However, in order to achieve better classification performance, these algorithms often introduce more parameters, led to an augmentation in the count of model parameters. In order to overcome this difficulty, we propose a graph neural network architecture search method based on multi-objective optimization algorithm, named MO-GNN. The algorithm sets the number of parameters and classification accuracy of the graph neural network model as the objective function, and combines the super network sharing weight strategy with the multi-objective algorithm to search the optimal graph neural network model and improve the search efficiency of the graph neural network architecture.

In summary, the contributions of this work are as follows:

1. In order to balance the number of parameters and classification accuracy, an algorithm called MO-GNN is proposed to combine multi-objective optimization with evolutionary algorithm to search for the optimal graph neural network model.

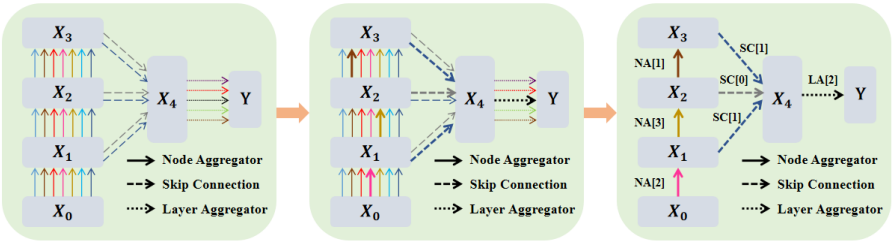


Fig. 1: The Overall Framework of Differential Method.

2. In order to improve the search efficiency, super network is combined with MO-GNN method by alternately optimizing hyperparameters with the model structure.

2 Related Works

2.1 Differential Graph Neural Architecture Search

The existing search strategies of graph neural network architecture include reinforcement learning and evolutionary algorithm. Reinforcement learning based methods include GraphNAS [10], Auto-GNN [16], which is time-consuming and inefficient. The differentiable method can search for a more expressive GNN model quickly and efficiently [11].

Fig. 1 shows the algorithm framework of the differential graph neural architecture search method.The training process of the differentiable algorithm is divided into two stages. In the first stage, the gradient descent method is used to train the network weights on the validation set of the super network, the weight information of each connection is obtained, and the operation with the maximum weight is used to replace the hybrid operation $\bar{o}^{ij}(x_i)$, so as to obtain a discrete network structure parameter α. In the second stage, the training set is used to optimize the network weight parameters under the condition of specific α.

In detail, a predefined set of fixed candidate operations $\mathcal{O}$ is defined. The set includes operation types such as node aggregation, layer aggregation, and skip connection. In the graph neural network model, $o(x_i)$ is defined as the input feature of node i. A directed edge (i, j) between node i and node j is associated with the operation $\bar{o}^{ij}(x_i)$ of the edge. Thus, the hybrid operation from node i to node j can be calculated as Equation 1:

$$\bar{o}^{ij}(x_i) = \sum_{o \in \mathcal{O}} \frac{\exp\left(\boldsymbol{\alpha}_o^{ij}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\boldsymbol{\alpha}_{o'}^{ij}\right)} o(x_i) \tag{1}$$

where $\boldsymbol{\alpha}_o^{ij}$ is represented as the type of operation between node i and node j. Therefore, the discrete search space can be converted to a continuous search space, replacing the discrete connection of the hybrid operation with $\bar{o}^{ij}(x_i)$ based on the weight of the hybrid operation.

Thus, we can get the final representation of the output node by calculating Equation 1 and inject it into different types of losses depending on the given task, a two-level optimization problem can be derived in Equation 2 and Equation 3:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{L}_{\text{val}} \ (\mathbf{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \tag{2}$$

$$\text{s.t. } \mathbf{w}^*(\boldsymbol{\alpha}) = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{tra}} \ (\mathbf{w}, \boldsymbol{\alpha}) \tag{3}$$

where $\mathcal{L}_{\text{tra}}$ and $\mathcal{L}_{\text{val}}$ can be represented as training loss and validation loss respectively, α and $\mathbf{w}^*(\boldsymbol{\alpha})$ are represented as network architecture and the corresponding weights after training phase.

2.2 Multi-objective Optimization

Compared with traditional single-objective optimization, multi-objective neural network architecture search focuses on a variety of different goals, such as model size, computational complexity, speed of operation, and interpretability, to achieve more comprehensive performance evaluation. In general, multi-objective neural network architecture search finds the best neural network structure by optimizing the weighted sum of multiple objective functions. In the field of neural network architecture search, Kim first proposed a multi-scale algorithm NEMO involving neural networks [18], used NSGA-II algorithm to minimize the error and inference time of the network, and searching the space of the number of output channels from each layer within the limited space of 7 different architectures. The main idea of the NSGA-II algorithm is to non-dominate sort each individual in the population according to its fitness, and select operations according to the sorting results to achieve the effect of multi-objective optimization [19].

Compared with the NSGA-II algorithm, MOPSO algorithm has fewer iterations and can find the optimal solution more quickly. Moreover, its convergence speed is relatively fast, and it takes a short time to converge to the optimal solution, while avoiding the occurrence of local optimal solution and greatly reducing the computational complexity in the training process.The innovation of the MOPSO algorithm mainly has two points: one is the use of an external repository and a grid-based particle distribution method to maintain the diversity of populations, where the external repository is storing the set of non-dominated solutions for each iteration [17]. The other innovation is the particle updation strategy. For multi-objective optimization problems, not only the convergence of the solution, but also the uniformity and breadth of the solution distribution should be considered. In order to ensure the diversity of the final solution, a new mutation strategy is taken to mutate the area of the particle distribution, and the mutation probability gradually decreases with the increase of evolutionary algebra .

3 Proposed Method

In this section, a multi-objective evolutionary algorithm based graph neural network architecture search framework MO-GNN is proposed. In order to improve the efficiency of the search stage and balance the performance of the model, multi-objective evolutionary algorithm which is combined with the super network is designed to search for the optimal graph neural network model.The overall framework of the algorithm is shown in Algorithm 1 and Fig 3.

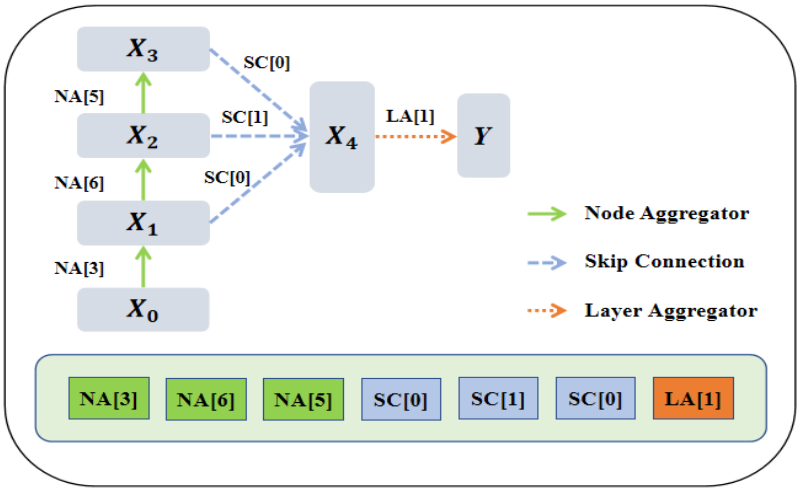


Fig. 2: Search Space Design and Particle Encode in MO-GNN Method

3.1 Search Space Construction

To build a well-performing search framework, the search space consists of three node aggregation layers and one layer aggregation layer. The node aggregation functions in the first three layers and the layer aggregation function in the last layer should be searched. The skip connections between the first three layers and the last layer also need to be selected. All candidate operations are shown in Table 1.

Among Fig. 2, X_0 and Y are represented as the input node and output node of the graph neural network model, respectively, and X_1, X_2, X_3, and X_4 are the four hidden nodes of the model. This network structure contains all candidate network models, also known as all particles in PSO. For all individual particles, the node aggregation operations, the layer aggregation operations, and the skip

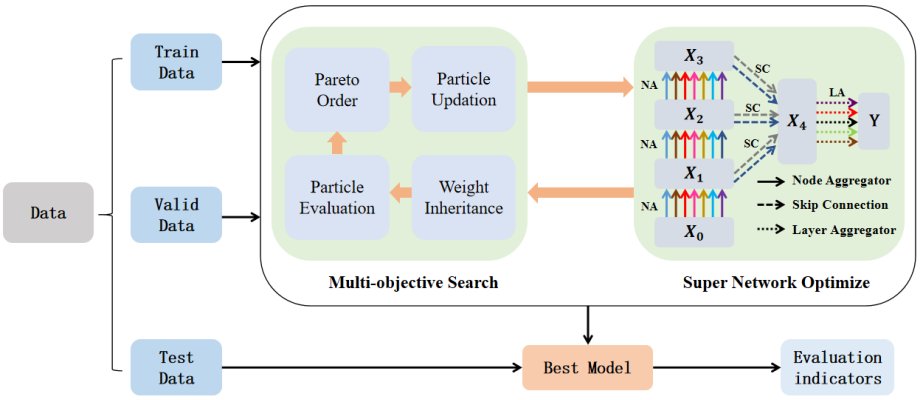


Fig. 3: The Overall Framework of MO-GNN Method

Table 1: The Operations of MO-GNN Method.

Types	Operations
Node Aggregator	Geniepath, Sage, Sage-sum, Sage-max, Gcn, Gin, Gat, Gat-sym, Gat-cos, Gat-Linear, Gat-Generalized-Linear
Skip Connection	Zero, Identity
Layer Aggregator	Concat, Max, Lstm

connection operations are be decoded by corresponding indices in the array. For example, the first, second, and third values of the particle array is represented as the indexes of the operation types between X_0 nodes and X_1 nodes, X_1 nodes and X_2 nodes, and X_2 nodes and X_3 nodes, respectively.

3.2 Search Strategy Design

In MO-GNN method, we focus on the two important indicators of the number of parameters and the overall classification accuracy of the graph neural network model. Since the number of model parameters and the overall classification accuracy are two unsolate indicators, usually we expect to design a graph neural network model with high overall classification accuracy and relatively small parameter quantity, how to comprehensively consider the relationship between the two and find the better graph neural network model is a problem that needs to be studied. Therefore, we construct a multi-objective optimization problem

Algorithm 1 MO-GNN Algorithm

Input: candidate operations(OP), particle swarm(S), particle swarm size (N), number of iterations ($iter$), training dataset (Tra_{data}), validation dataset (Val_{data}), archive set (A), archieve size (n), fitness (F_1, F_2) and architecture ($\boldsymbol{\alpha}$)
Output: The best GNN architecture

```
S = [P_1.....P_N]← initialize the particle swarm
A = [A_0.....A_n]← archive set includes dominated solutions
while i < iter do
    Optimize the improved SuperNet by gradient descent on Tra_data

  while j < N do
      Update pBest by comparing F_1(x_j), F_2(x_j)
      Update the A[i]
      Update gBest by minmum crowding distance
      Update the particles
  end while
    F_1, F_2, α ← Pareto solutions with balanced F_1 and F_2 of the architecture
  during iter on Val_data
end while
return F_1, F_2, α
```

consisting of two objective functions of classification accuracy and the number of model parameters, which can be expressed as Equation 4 and Equation 5:

$$F_1 = \frac{TN + TP}{TP + TN + FP + FN} \tag{4}$$

$$F_2 = C_o \times (k_w \times k_h \times C_i + 1) \tag{5}$$

Among the equation, F_1 represents the overall classification accuracy of the graph neural network model, (TP + TN) represents the correct prediction sample, (TP+ TN + FP + FN) represents all samples. F_2 represents the number of parameters of the graph neural network model, C_o represents the number of output channels, C_i represents the number of input channels, k_w represents the convolution kernel width, and k_h represents the convolution kernel height.

For the weight training of different particles or subnetworks of particle swarm algorithm, it is repetitive and redundant, which requires a lot of training time and computer resources. So the multi-objective particle swarm algorithm combined with supernet is proposed. In MO-GNN method, the weights of all candidate network architectures are obtained by training the weights of the super network once in the training process, which avoids training the weights of each particle or subnetwork separately, and realizes weight sharing in the search process of neural network architecture. In this algorithm, the supernet is built by connecting all candidate operation types with nodes. The weight of the supernet contains the weights of all candidate particles or networks, so all particles can inherit the corresponding weights from the supernet. In the process of each iteration,

the supernet is first trained with the training set, and all candidate particles inherit the corresponding weights in the super network, and then the MO-GNN algorithm is used to search for the best particles among all candidate particles, so as to realize the combination of particle swarm algorithm and super network, thereby reducing the training time and calculation amount in the model search process.

In the process of particle updation, the particle is optimized by comparing with the gBest and the pBest and randomly selecting. According to the randomly generated number C between [0,1], compare with the parameter Cg we set, and if C is not less than Cg, the difference between the particle gBest and the particle is taken; If C is less than Cg, take the difference beteeen the particle pBest and the particle. If there is no difference, it means that the two are the same and the current particle is not updated; else it means that the architecture in pBest or gBest is better than the particle architecture to be updated, and the particles selected from pBest or gBest are selected to replace the current solution according to the probability.

4 Experimental Result

4.1 Experimental Settings

In this section, we describe the experimental setup and parameters of the algorithm, and demonstrate the advantages of the proposed MO-GNN algorithm in transduction and induction tasks through a number of experiments. The experimental conditions are as follows: i9-10900 CPU, NVIDIA GTX1080 Ti GPU, 48GB memory. In addition, we set the population size to 100, the number of iterations to 40, the archive threshold to 10, and the mutation parameter Cg to 0.5. In all methods, the learning rates of the weights during search and testing are 0.025 and 0.05, respectively. All information about the datasets we use is shown in Table 2.

A. Transduction Learning

Transduction learning is a method of predicting specific test set data by observing specific training set data, which can use the information of unlabeled test set data to find clusters and classify more effectively. We employ three benchmark reference network datasets, including Cora, Citeseer, and Pubmed, all of which are citation networks provided [20] for transduction node representation learning. The Cora dataset contains 2708 research papers divided into 7 machine learning classes, such as reinforcement learning and genetic algorithms. There are 5429 edges between them, and each paper node is described by a 1433-dimensional eigenvector. The Citeseer dataset contains 3327 research papers of 6 classes with 4732 links between them, and each paper node has a 3703-dimensional feature vector. The Pubmed dataset contains 19,717 document nodes and 44,338 edges. Each node belongs to one of 3 classes and has a 500-dimensional feature vector.

For these three datasets, all nodes in the graph are divided into training set, validation set, and test set in the proportions of 60%, 20%, and 20%.

B. Inductive Learning

Inductive learning refers to inference from specific observed data to general data, using labeled data for model training, but its training set data does not contain test set data, and then using the trained model to predict the labels of the test set data. For inductive learning, training graphs and test graphs are different. Therefore, unlike transduction learning, inductive learning involves embedding learning of several different subgraphs. We use a protein-protein interaction (PPI) dataset containing 24 subplots for inductive node representation learning. PPI consists of graphs corresponding to different human tissues, with a total of 56,944 nodes and 818,716 edges. Each graph has an average of 2372 nodes, each node has 50 features, including positional gene sets, baseline gene sets, and immune features. Each node corresponds to multiple classes (labels) from 121 classes. For this dataset, we use 20 graphs for training, 2 graphs for validation, and 2 graphs for testing, where neither the validation graph nor the test graph are related to the graphs in the training set.

For the above dataset, we follow [21], using a dataset of 30% for training, 10% for validation, and the remaining 60% for testing.

Table 2: The Comparison of Different Datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
PubMed	3327	4552	3703	6
CiteSeer	19717	44324	500	3
PPI	56944	818716	50	121

For the datasets Cora, CiteSeer, and PubMed, each class utilizes 20 nodes for the training set, 500 nodes for the validation set, and 1000 nodes for the test set. As for the PPI dataset, this paper employs 20 graphs for training, 2 graphs for validation, and 2 graphs for testing, where the validation and test graphs are disconnected from the graphs in the training set.

4.2 Comparison with Other Methods

A. Introduction to Comparison Algorithms

In order to evaluate the performance of our proposed method, the most advanced artificially designed graph neural network model and the latest method of graph neural network architecture search methods are selected for comparison.In order to fairly compare the efficiency and performance of the manually designed graph neural network model and the graph neural network architecture search method, the number of layers of all graph neural network models are set to 3.

For all comparison algorithms, we report the average best test accuracy and standard deviation for the architecture searched. Specifically, we retrain each search schema ten times to avoid randomness.

GCN [2]: The paper expands the convolutional range from the traditional Euclidean space to the non-Euclidean space, and can perform convolution operations on the structural data of the non-Euclidean space such as graphs, and the framework proposes a local first-order approximation of spectral convolution based on the frequency domain, and introduces convolution into the graph neural network model.

GAT [4]: The framework utilizes a self-attention mechanism to solve the shortcomings of previous methods based on graph convolution or its approximation, by superimposing layers of nodes capable of participating in their neighborhood features, we can assign different weights to different nodes in the neighborhood without the need for any expensive matrix operations (such as inverse) or prior knowledge of the graph structure.

GraphSAGE [14]: The author proposes an inductive learning framework to extend graph convolution to inductive learning tasks by sampling neighbor nodes and training the function of aggregating node neighbors, and generalizing the unknown nodes.

LGCN [22]: This framework is to deal with the problem that the number and order of neighbors in the graph structure are not fixed, and automatically select a fixed number of neighbor nodes for each feature, transform the graph structure data into a one-dimensional grid structure, and apply conventional convolution operations on the graph.

GraphNAS [10]: The framework uses recurrent networks to generate variable-length strings describing graph neural network architectures and reinforcement learning to train recurrent networks to maximize the expected accuracy of the architecture generated on validation datasets.

SNAG [23]: To compensate for the deficiencies in search space, expressiveness, and search efficiency of GraphNAS and Auto-GNN, the authors propose the SNAG framework (Simplified Neural Network for Neural Structure Search Graphs), including a new search space and reinforcement learning-based search algorithms.

Genetic-GNN [13]: The author proposes a new evolutionary algorithm framework that dynamically approaches the optimal fit of each other in the process of alternating evolution between GNN model structures and hyperparameters.

SANE [11]: This author proposes a framework that attempts to search aggregation neighborhoods to automate the design of data-specific GNN schemas. By designing a novel and expressive search space, a differentiable search algorithm is proposed, which is more efficient than previous reinforcement learning-based search algorithms.

B. The Analysis of Accuracy and Parameters

In this section, the results compared with other methods in terms of classification accuracy and parameters are shown and analyzed. To illustrate the feasibility of MO-GNN method, the convergence of the model is also studied.

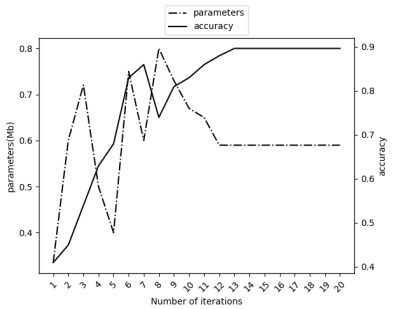
Table 3 shows the classification results of transduction learning and inductive learning tasks, from which it can be seen that the overall classification accuracy of the graph neural network architecture discovered by MO-GNN is significantly better than that of the manually designed graph neural network model. For the transduction and induction tasks, it can be seen that the overall classification accuracy of MO-GNN on the three datasets is significantly higher than that of manually designed graph neural network models such as GCN and GAT. Therefore, we mainly compare the MO-GNN algorithm with the SOTA model of the NAS method.

Table 3: The comparison of MO-GNN with Other Methods.

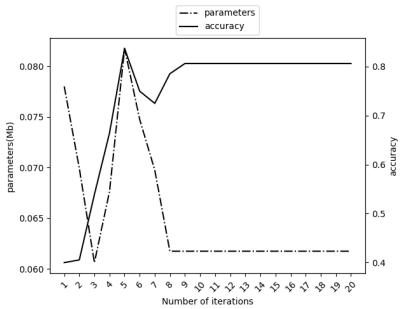
	Cora		PubMed		CiteSeer		PPI	
	Accuracy	Parameters	Accuracy	Parameters	Accuracy	Parameters	Accuracy	Parameters
GCN [2]	85.7±0.3%	**25.9K**	73.6±0.4%	**26.2K**	87.9±0.1%	**50.8K**	73.5±0.2%	**182.7K**
GAT [4]	86.1±0.2%	108.2K	74.0±0.2%	48.2K	87.7±0.3%	296.5K	95.3±0.4%	893.2K
GraphSAGE [14]	85.9±0.4%	68.2K	74.1±0.6%	54.3K	87.4±0.2%	58.9K	65.7±0.6%	225.3K
LGCN [22]	85.9±0.3%	69.5K	74.3±0.5%	73.1K	87.2±0.4%	65.3K	78.4±0.2%	227.3K
GraphNAS [10]	86.7±0.4%	98.5K	77.0±0.3%	83.5K	87.9±0.5%	245.7K	96.9±0.1%	1095.1K
SNAG [23]	86.9±0.6%	76.3K	76.5±0.6%	69.1K	87.7±0.4%	220.9K	97.6±0.1%	823.7K
Genetic-GNN [13]	87.1±0.3%	75.7K	77.8±0.5%	142.5K	88.6±0.6%	190.4K	98.0±0.1%	859.4K
SANE [11]	87.3±0.6%	86.1K	77.5±0.6%	74.9K	88.4±0.2%	251.3K	98.1±0.1%	964.3K
MO-GNN	**88.7±0.3%**	59.0K	**78.8±0.4%**	61.8K	**89.7±0.1%**	59.1K	**98.6±0.2%**	649.5K

For the Cora dataset, compared with the GraphNAS [10], SNAG, Genetic-GNN [13] and SANE [11] algorithms, the MO-GNN method improves the overall classification accuracy of the graph neural network model by 2%, 1.8%, 1.6% and 1.4%, respectively. Meanwhile, the model also has certain advantages in terms of the number of parameters, which was reduced by 39.5K, 17.3K, 16.7K and 27.1K respectively compared with the above methods. For the PubMed dataset, the MO-GNN algorithm improves the classification accuracy of the graph neural network model by 1.8%, 2.3%, 1.0% and 1.3%. Compared with Genetic-GNN, the number of parameters is reduced by 56.6%. For the CiteSeer dataset, MO-GNN method still maintain the same advantages as other two homogeneous datasets. For induction tasks, similar to transduction tasks, MO-GNN consistently outperforms all baselines, compared with the GraphNAS [10], SNAG, Genetic-GNN [13] and SANE [11] algorithms, MO-GNN algorithm improves the model classification accuracy by 1.7%, 1%, 0.6% and 0.5%, respectively. Compared with the GraphNAS algorithm, the number of parameters was reduced by 40.7% in PPI dataset. In MO-GNN algorithm, the increase of model classification accuracy does not lead to a significant increase in the number of parameters, and even greatly reduces the number of parameters of the graph neural network model.

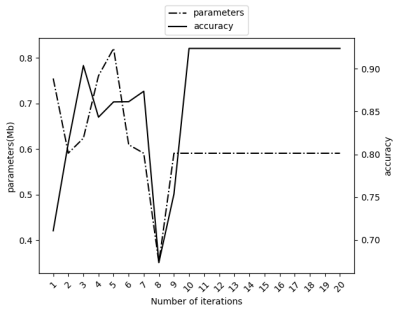
In general, compared with the manually designed graph neural network model and the most advanced graph neural network architecture search method, the proposed MO-GNN algorithm has greatly improved the classification accuracy and parameter quantity.



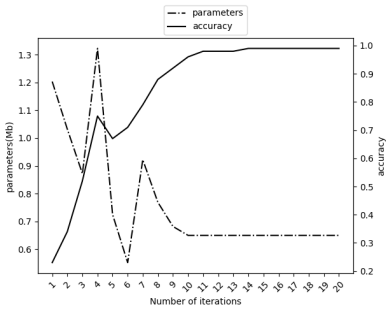
(a) The parameter quantity and classification accuracy of Cora



(b) The parameter quantity and classification accuracy of PubMed



(c) The parameter quantity and classification accuracy of CiteSeer



(d) The parameter quantity and classification accuracy of PPI

Fig. 4: Comparison of parameter quantity and classification accuracy for different datasets

Fig. 4a-4d shows the change of classification accuracy and parameter quantity of the optimal particle verification set of the MO-GNN algorithm in the iterative process. Under ideal conditions, we would choose a graph neural network model with higher accuracy and fewer parameters. The number of model parameters is related to the complexity of the model, and fewer model parameters means lower the complexity of the model. After rapid changes, the model gradually stabilizes and remains unchanged in the final iteration. It can be seen from the figure that in the first few iterations, the amount of parameters of the model has changed dramatically, for the Cora dataset, the amount of model parameters of the optimal particles converges at the 13th iteration, but the classification accuracy of the optimal particles in the validation set still increases after the convergence of

the model parameter amount, mainly because the weight parameters inherited from the supernet continue to be optimized during subsequent iterations until the end of the iteration. For the PubMed dataset, the number of parameters and classification accuracy converged on the 9th iteration. The two objective functions of the CiteSeer dataset and the PPI dataset converge at the 10th and 14th iterations, respectively.

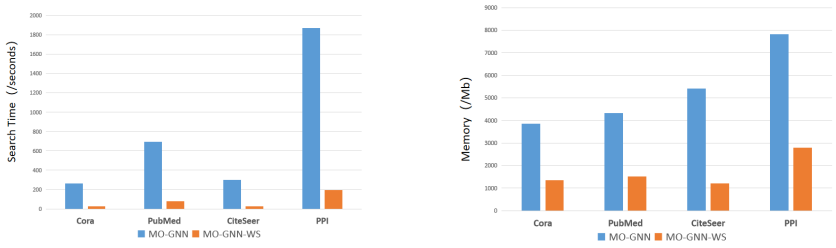
From the above convergence analysis, we can see that MO-GNN converges within 14 iterations on the four datasets of the transduction and induction task.

4.3 Ablation Experiments

In this section, the effect of different number of the layers and the validity of the super network is analyzed and verified.

It can be seen from Table 3 that the classification performance of shallow graph neural network models is relatively average, mainly because the model with simple structure may not be able to extract deep features, and the deep graph neural network model does not show better classification performance, which may be due to the problem of overfitting in the case of limited training samples.

In order to further illustrate the effectiveness of our proposed supernet sharing weight strategy, the algorithm using the super network sharing weight strategy is named as MO-GNN-WS, and the algorithm without the strategy is represented as MO-GNN. In Fig. 5a-5b, it is clear to seen that compared with MO-GNN, the time loss in the search phase on the four graph datasets is reduced by about 90%. In terms of memory loss,the weight sharing strategy reduced by 65.0%, 64.8%, 77.6% and 63.2%, respectively. In general, the proposed super network sharing weight strategy greatly reduces the resource consumption of graph neural network architecture search.



(a) The Comparison of Search Time in MO-GNN and MO-GNN-WS

(b) The Comparison of Memory in MO-GNN and MO-GNN-WS

Fig. 5: Comparison of time and memory in MO-GNN and MO-GNN-WS

5 Conclusion

In the paper, MO-GNN, a graph neural network search model based on multi-objective particle swarm optimization algorithm is proposed for the first time.

Extensive experiments have been carried out on four real datasets of the transformation and induction task. Compared with the most advanced graph neural network architecture search method and the manually designed graph neural network model, the experimental results show that the model has a great improvement in classification accuracy. From the perspective of model complexity, compared with the existing algorithms, the graph neural network model searched by the framework significantly reduces the number of parameters of the model. Combined with the super network, our method greatly improves the search efficiency and decrease the training stage.

References

1. Scarselli F, Gori M, Tsoi A C, Hagenbuchner M and Monfardini G. The Graph Neural Network Model[J]. IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80, 2008.
2. Chen M, Wei Z, Huang Z, et al. Simple and deep graph convolutional networks[C]. International Conference on Machine Learning, pp. 1725-1735, 2020.
3. He Z, Zhong Y and Pan J. Emotion-related awareness detection for patients with disorders of consciousness via graph isomorphic network[C]. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 3158-3164, 2022.
4. Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks[J]. Stat, vol. 1050, no. 20, pp. 10-22, 2017.
5. Ranjan E, Sanyal S, Talukdar P. Asap: Adaptive structure aware pooling for learning hierarchical graph representations[C]. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no.4, pp. 5470-5477, 2020.
6. Qi X, Liao R, Jia J, et al. 3d graph neural networks for rgbd semantic segmentation[C]. Proceedings of the IEEE International Conference on Computer Vision, pp. 5199-5208, 2017.
7. Huang C, Xu H, Xu Y, et al. Knowledge-aware coupled graph neural network for social recommendation[C]. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no.5, pp. 4115-4122, 2021.
8. Song W, Xiao Z, Wang Y, et al. Session-based social recommendation via dynamic graph attention networks[C]. Proceedings of the Twelfth ACM international Conference on Web Search and Data Mining, pp. 555-563, 2019.
9. Feng G, Wang H, Wang C. Search for deep graph neural networks[J]. Information Sciences, 649: 119617, 2023.
10. Gao Y, Yang H, Zhang P, et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning[J]. ArXiv:1904.09981, 2019.
11. Zhao H, Yao Q, Weiwei T U. Search to aggregate neighborhood for graph neural network[C]. 2021 IEEE 37th International Conference on Data Engineering, pp. 552-563, 2021.
12. Zhang X, Ma H, Zhang X, et al. Compact model-free adaptive control algorithm for discrete-time nonlinear systems[J]. IEEE Access, 2019, 7: 141062-141071.

13. Shi M, Tang Y, Zhu X, et al. Genetic-GNN: Evolutionary architecture search for Graph Neural Networks[J]. Knowledge-based Systems, vol. 247, pp.108752, 2022.
14. Oh J, Cho K, Bruna J. Advancing graphsage with a data-driven node sampling[J]. ArXiv:1904.12935, 2019.
15. Qin Y, Wang X, Zhang Z, et al. Multi-task graph neural architecture search with task-aware collaboration and curriculum[J]. Advances in Neural Information Processing Systems, 36, 2024.
16. Zhou K, Song Q, Huang X, et al. Auto-GNN: Neural Architecture Search of Graph Neural Networks[J]. Frontiers in Big Data, vol. 5, pp. 1029307, 2022.
17. Liang J, Ban X, Yu K, et al. A survey on evolutionary constrained multi-objective optimization[J]. IEEE Transactions on Evolutionary Computation, vol. 27, no. 1, pp. 201-221, 2022.
18. Y. Kim, B. Reddy, and S. Yun. NEMO: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy[C]. International Conference on Machine Learning, vol. 1, pp. 1-8, 2017.
19. Wang X, Wang X, Jin L, et al. Evolutionary algorithm-based and network architecture search-enabled multiobjective traffic classification[J]. IEEE Access, vol.9, pp. 52310-52325, 2021.
20. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, and Eliassi Rad T. Collective Classification in Network Data[J]. AI Magazine, vol. 29, no. 3, pp. 93, 2008.
21. Wang Z, Lv O, Lan X, and Zhang Y. Cross-lingual knowledge graph alignment via graph convolutional networks[C]. Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 349-357, 2018.
22. Gao H, Wang Z, Ji S. Large-Scale Learnable Graph Convolutional Networks[C]. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1416-1424, 2018.
23. Zhao H, Wei L, Yao Q. Simplifying Architecture Search for Graph Neural Network[J]. ArXiv: 2008.11652, 2020.