Automatic Graph Learning with Evolutionary Algorithms: An Experimental Study

Chenyang Bu, Yi Lu, and Fei Liu(✉)

School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei, China
chenyangbu@hfut.edu.cn, {luyi,feiliu}@mail.hfut.edu.cn

Abstract. In recent years, automated machine learning (AutoML) has received widespread attention from academia and industry owing to its ability to significantly reduce the threshold and labor cost of machine learning. It has demonstrated its powerful functions in hyperparameter optimization, model selection, neural network search, and feature engineering. Most AutoML frameworks are not specifically designed to process graph data. That is, in most AutoML tools, only traditional neural networks are integrated without using a graph neural network (GNN). Although traditional neural networks have achieved great success, GNNs have more advantages in processing non-Euclidean data (e.g., graph data) and have gained popularity in recent years. However, to the best of our knowledge, there is currently only one open-source AutoML framework for graph learning, i.e., AutoGL. For the AutoGL framework, traditional AutoML optimization algorithms such as grid search, random search, and Bayesian optimization are used to optimize the hyperparameters. Because each type of traditional optimization algorithm has its own advantages and disadvantages, more options are required. This study analyzes the performance of different evolutionary algorithms (EAs) on AutoGL through experiments. The experimental results show that EAs could be an effective alternative to the hyperparameter optimization of GNN.

Keywords: Automatic graph learning · Evolutionary algorithms · AutoML

1 Introduction

Automated machine learning (AutoML) [1–3] refers to the automation of the entire machine learning process from model construction to application. Compared with traditional machine learning, AutoML can achieve results equivalent

C. Bu—Was partly supported by the National Natural Science Foundation of China (No. 61806065 and No. 91746209), the Fundamental Research Funds for the Central Universities (No. JZ2020HGQA0186), and the Project funded by the China Postdoctoral Science Foundation (No. 2018M630704).

D. N. Pham et al. (Eds.): PRICAI 2021, LNAI 13031, pp. 513–526, 2021.
https://doi.org/10.1007/978-3-030-89188-6_38

to or better than human experts with no or few human interventions. Therefore, AutoML can lower the threshold of algorithm learning and use; thus, it can be helpful for applications in machine learning algorithms in real scenarios.

Most of the existing AutoML research and frameworks do not consider the particularity of graph data, therefore they cannot be directly applied to graph machine learning models [4]. Many research problems in different fields can be naturally modeled into graph machine learning, such as social media analysis [5], recommendation systems [6], and protein modeling. Although traditional deep learning methods have achieved great success, their performance in processing non-Euclidean spatial data may be unsatisfactory [7,8]. For example, in e-commerce, a graph-based learning system can use the interaction between users and products to make very accurate recommendations, but the complexity of graphs makes existing deep learning algorithms face huge challenges [6]. This is because the graph is irregular; that is, each graph has an unordered node with a variable size, and each node in the graph has a different number of adjacent nodes. The characteristics of graph data lead to some important operations (such as convolution) that are easy to calculate in image processing no longer suitable for direct use in graph data [7]. In addition, a core assumption of existing deep learning algorithms is that data samples are independent of each other [7]. For graphs, this is not the case. That is, each data sample (i.e., a node) in the graph has edges related to the other nodes. This information can be used to capture the interdependence between instances [7]. Therefore, graph neural networks (GNNs) [9] have been proposed to specifically process graph data. Compared with traditional neural networks, GNNs have better reported experimental results in processing graph data. Thus, this research line has received widespread attention recently [10]. Typical GNNs include graph convolution networks (GCNs) [11] and graph attention networks (GAT) [12].

Automatic graph learning (AutoGL) refers to AutoML for graph data. According to [4], AutoGL is the first open-source automatic graph learning toolkit[1] that was released on December 21, 2020 [4]. This tool supports fully automatic machine learning on graph data, implements typical GNNs including GCN and GAT, and supports the two most common tasks in graph machine learning, that is, node classification and graph classification. Hyperparameter optimization (HPO) algorithms mainly include grid search, random search, simulated annealing, and Bayesian optimization methods. However, each HPO algorithm adopted in AutoGL has its own disadvantages. For example, grid search is not efficient, because it tries every possible combination of hyperparameters. Random search generally performs better than grid search in previously reported results; however, the obtained results may still not be good in the case of limited computing resources. Simulated annealing uses a single individual for optimization, leading to the final solution having a certain dependence on the initial solution. Gradient descent requires gradient information, which typically cannot be obtained for the model training process because only the input and output of

[1] https://github.com/THUMNLab/AutoGL.

the models are available during the tuning process. Therefore, we require more HPO options for the AutoGL.

In this study, we attempted to apply evolutionary algorithms (EAs) to AutoGL, as a class of nature-inspired population-based stochastic search algorithms. Because searching is based on a population of candidate solutions, instead of a single one, EAs have good adaptability and robustness [13]. Moreover, no or few restrictions are required for cost functions, therefore EAs are suitable for solving complex problems that cannot be solved well by traditional methods [14,15]. Therefore, EAs have been widely used to solve various optimization problems in practical applications [16–19].

Researchers may be concerned about two research questions concerning the application of EAs in automatic graph learning. The first question is whether EAs perform better than other traditional HPOs for AutoGL. Second, EAs are so diverse that researchers may find it difficult to choose which EA to apply in their own experiments, or they may need to spend an unexpected time to compare.

This study focuses on analyzing the above two research questions through experiments. Both the topology and learning parameters of GNNs were encoded into chromosomes for simultaneous optimization using different EAs. Several EAs have been applied, and a number of experiments on two real-world datasets have been performed. To the best of our knowledge, this is the first study to compare the experimental performance of various EAs for AutoGL. The experimental results show that EAs could be an effective alternative to the HPO of AutoGL.

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

1. To the best of our knowledge, this is the first study to apply EAs to the AutoGL framework, where AutoGL is the first open-source AutoML framework for graph learning.
2. An experimental analysis was conducted of the performance of different EAs for AutoGL through extensive experiments. This study may have reference value for researchers attempting EAs in this field.

The rest of this paper is organized as follows. Details of the algorithm is given in Sect. 2. The experiments are discussed in Sect. 3, and Sect. 4 concludes the paper.

2 Approach

In this section, we introduce the motivation of this study and the specific operation process of the method mentioned.

2.1 Motivation

The open-source toolkit AutoGL [4] shares various HPO algorithms for GNNs, such as grid search, random search, simulated annealing, and Bayesian optimization methods. However, there are disadvantages detailed in Fig. 1. Therefore, we

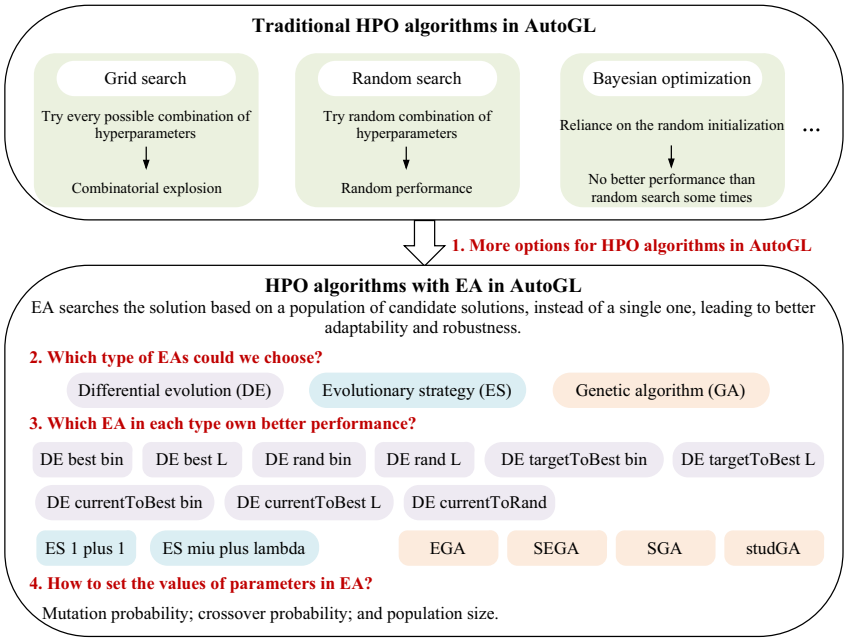

Fig. 1. The motivation of this study to present an experimental study of AutoGL with EA. The main task of the study is to solve the four points.

attempt to show more options for HPO algorithms for GNNs, i.e., AutoGL with EA algorithms.

EA has been widely used to solve various optimization problems [20,21]. In this study, EA is applied to HPO for GNNs to attempt to address the above disadvantages of the existing HPO algorithms, as reflected in the following aspects.

- Because some existing HPO algorithms (e.g., simulated annealing and Bayesian optimization methods) use a single individual for optimization, the final solution has a certain dependence on the initial solution. EA is introduced into HPO algorithms because it searches the solution based on a population of candidate solutions, instead of a single one, owing to its good adaptability and robustness [13,14].
- Some existing HPO algorithms (e.g., grid search and random search) are not efficient when computing resources are limited. HPO algorithms using EA obtain high-quality individuals through population evolution based on heuristic information.

To facilitate researchers to choose AutoGL with EA in applications, the following research points are analyzed in the experimental study, as shown in Fig. 1: 1) More options for the HPO algorithm are displayed. 2) Various types of EAs are compared. 3) EAs in each type are compared. 4) Different values of parameters in EA are set to analyze the performance.

Table 1. Search space of discrete parameters

Parameter name	Type	Min	Max	Scaling type
Learning Rate $\mathcal{P}_1$	float	0.01	0.05	Log
Weight Decay Rate $\mathcal{P}_2$	float	0.0001	0.001	Log
Dropout Rate $\mathcal{P}_3$	float	0.2	0.8	Linear

Table 2. Search space of continuous parameters

Parameter name	Searching space
Number of Hidden Units $\mathcal{H}_1$	{4, 5,..., 16}
Number of Attention Heads $\mathcal{H}_2$	{6, 8, 10, 12}
Activation Function $\mathcal{P}_4$	{leaky relu, relu, elu, tanh}
Max Epoch $\mathcal{P}_5$	{100, 101,..., 300}
Early Stopping Round $\mathcal{P}_6$	{10, 11,..., 30}

In this section, AutoGL with EA is modeled (corresponding to point 1). Then, the experimental results are analyzed from the aspects of the abovementioned points 2 to 4.

2.2 Optimized Parameters

In this subsection, AutoGL with EA is used to optimize the hyperparameters of two typical GNN models, i.e., GCN and GAT.

In GCN, the number of layers in the convolution structure was fixed, and only the number of units in the hidden layer ($\mathcal{H}_1$) was adjusted. It is because when the number of layers is above two, the effect is not greatly improved, and when the number of layers is too high, the training effect is significantly reduced [11]. In GAT, the number of nodes in the hidden layer ($\mathcal{H}_1$) of the model and the number of multi-head-attentions ($\mathcal{H}_2$) in the GAT model participated in the optimization of the model, and the same number of hidden layers was fixed.

In addition to the abovementioned hyperparameters, the common hyperparameters of GCN and GAT are the learning rate ($\mathcal{P}_1$), weight decay rate ($\mathcal{P}_2$), dropout rate ($\mathcal{P}_3$), activation function ($\mathcal{P}_4$), maximum epoch ($\mathcal{P}_5$) and early stopping round ($\mathcal{P}_6$).

The parameters involved in the optimization are discrete and continuous, and information such as the range of values for continuous parameters is presented in Table 1. The range of the discrete parameters is listed in Table 2.

2.3 Encoding

The encoding methods of the hyperparameters in GCN and GAT are given in this subsection.

The encoding process contains the following two parts: the encoding of the topological structure of the network and the encoding of the hyperparameters

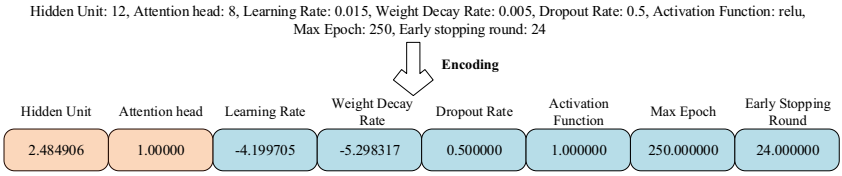

Fig. 2. An example encode of solution

in the training process, where the former is embodied in the coding of the number of hidden units and attention heads. To better participate in the search in the EA, the data types of each parameter were re-coded and the upper and lower limits were set; discrete parameters, such as categories and integers, were re-coded as floating-point numbers to facilitate the operation of the EA, and some parameters were logged before they participated in the optimization of the EA. At each evaluation, it was converted from the floating-point number to the original data type through a certain decoder and within the valid range.

For GCN model, chromosome is represented as Eq. (1).

$$f = \{\mathcal{H}_1^*, \mathcal{P}_1^*, \mathcal{P}_2^*, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5, \mathcal{P}_6\}\,. \tag{1}$$

For GAN model, chromosome is represented as Eq. (2).

$$f = \{\mathcal{H}_1^*, \mathcal{H}_2^*, \mathcal{P}_1^*, \mathcal{P}_2^*, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5, \mathcal{P}_6\}\,, \tag{2}$$

where $\mathcal{H}_1^*$ represents the number of hidden units $\mathcal{H}_1$ transformed as $\ln \mathcal{H}_1$. $\mathcal{H}_2^*$ represents the mapping of the value $\mathcal{H}_2$ in the discrete value space to the represented ordinal number. $\mathcal{P}_1^*$ represents the number of units $\mathcal{P}_1$ transformed as $\ln \mathcal{P}_1$. $\mathcal{P}_2^*$ represents the number of unit $\mathcal{P}_2$ been transformed as $\ln \mathcal{P}_2$. $\mathcal{P}_4^*$ represents the mapping of the value $\mathcal{P}_4$ in the discrete value space to the represented ordinal number. f represents the individuals of this set of hyperparameters.

For example, a set of hyperparameters are as follows, the number of hidden units: 12, the number of attention heads: 8, learning rate: 0.015, weight decay rate: 0.005, dropout rate: 0.5, activation function: relu, max epoch: 250, and early stopping round: 24. They are encoded as (2.484906, 1.00000, –4.199705, –5.298317, 0.500000, 1.000000, 250.000000, 24.000000), for a more intuitive view, referred to Fig. 2.

2.4 Parameter Evolution

The objective of HPO algorithms is to optimize the individual fitness, where the individual represents the encoded chromosome regarding the hyperparameters. The following equations are used to evaluate the fitness of individual as shown in Eq. (3).

$$fitness(I_i) = loss(Model_{opt}(i)), \tag{3}$$

Table 3. Dataset statistics [22]

Dataset	Classes	Nodes	Edges	Features
Cora	7	2708	5429	1433
Citeseer	6	3327	4732	3703

where I_i denotes the individual. $fitness(I_i)$ denotes the fitness function of individual I_i. $loss()$ denotes the loss function, e.g., accuracy and logloss. $Model_{opt}(i)$ is shown in Eq. (4).

$$Model_{opt}(i) = model.train(decode(f_i)), \tag{4}$$

where $Model_{opt}(i)$ denotes the optimized model after the training process using the hyperparameter setting I_i. f_i denotes the encoded chromosome according to Eq. (2). $decode(f_i)$ denotes the reverse process of encoding, reversing all the values in the gene nodes into the original format to facilitate the construction of the model.

3 Experiment

In this section, the performance of HPO with EAs is experimentally investigated in the following aspects. 1) The performance between the HPO with EA and the traditional HPO algorithm is compared (Sect. 3.2) on different tasks, i.e., GCN and GAT. 2) The performance comparison between the HPO with various EAs is analyzed (Sect. 3.3). 3) The effects of different parameter settings in the EAs on the results are shown, including the mutation probability, the crossover probability, and the population size (Sect. 3.4).

All experiments were performed on a PC with an Intel(R) Xeon(R) Gold 6151 CPU, 32 GB memory, and a GeForce RTX 2080 TI GPU. The implementation of genetic algorithms relies on the open-source library[2].

3.1 Setup

The setup is detailed in the subsection, including datasets, baselines, and metrics.

Datasets. In the experiment, two famous datasets, i.e., Cora and Citeseer are used, where the dataset statistics [22] are presented in Table 3.

Baselines. The HPO models using various EAs were compared with the traditional HPO models with the same number of evaluations. And the parameter setting in the experiment for EAs is shown in Table 4. The parameters are not finely turned.

The baselines are the traditional HPO methods, shown as follows.

[2] https://github.com/geatpy-dev/geatpy.

Table 4. Parameter setting

Population size	Mutation rate	Crossover rate	Evaluation rounds
100	0.2	0.7	10000

- GCN-Random search. Random search which conducted on the GCN task.
- GAT-Random search. Random search which conducted on the GAT task.
- GCN-Bayesian optimization. Bayesian optimization which conducted on the GCN task.
- GCN-Bayesian optimization. Bayesian optimization which conducted on the GAT task.
- GCN-Simulated annealing. Simulated annealing (SA) which conducted on the GCN task.
- GCN-Simulated annealing. SA which conducted on the GAT task.

The HPO with EAs applied three types of EAs, i.e., the differential evolution (DE) algorithms, the evolutionary strategy (ES) algorithms, and the genetic algorithms (GA). For a more comprehensive comparison, various algorithms are chosen in each type of EAs, listed as follows. The same as the baselines, these EAs are also conducted on the GCN and GAT tasks.

- DE: DE best bin [23], DE best L [23], DE rand bin [23], DE rand L [24], DE targetToBest bin [25], DE targetToBest L [25], DE currentToBest bin [26], DE currentToBest L [26], and DE currentToRand [26].
- EA: ES 1 plus 1 [27] and ES miu plus lambda [27].
- GA: EGA, SGA, SEGA [28], and studGA [29].

Metrics. The accuracy is used to quantify the performance from the classification perspective, shown as follows. Larger values indicate a better performance.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

where TP, FP, TN, FN denote the numbers of true positives, false positives, true negatives, and false negatives, respectively.

3.2 Comparison Between HPO with EA and Traditional HPO

Different types of EAs share the same parameter setting (detailed in Table 4) to conduct the GCN and GAT tasks. Note that the results are the average values from 10 independent tests to avoid accidental results.

As shown in Fig. 3, they have different performances for the tasks, however, the performance of HPO with EA is better than those of the traditional HPO methods in most cases. It demonstrated that the HPO with EA is a good option for AutoGL. The detailed results are displayed in Table 5. It can be found

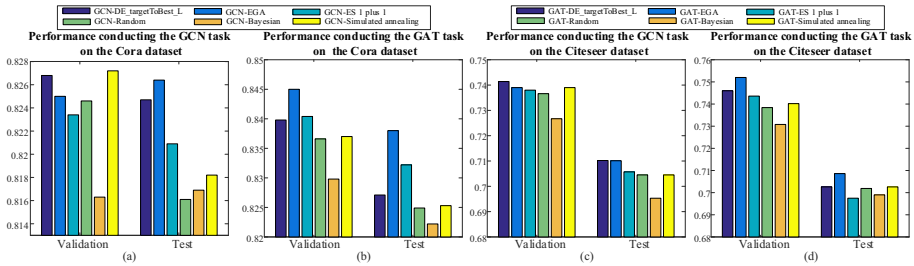

Fig. 3. Performance comparison between HPO with EA and traditional HPO conducting the GCN and GAT tasks on the Core and Citeseer datasets. The ordinate of each subgraph is the accuracy (the higher, the better). It demonstrated that the HPO with EA is a good option for AutoGL.

Table 5. Accuracy performance results among the various types of EAs

Method	Cora		Cora	
	Average (validation)	Standard deviation (validation)	Average (test)	Standard deviation (test)
GCN-DE_targetToBest_1_L	0.8268	0.0022	0.8247	0.0086
GCN-EGA	**0.825**	0.0037	**0.8264**	0.0093
GCN-ES_1_PLUS_1	0.8234	**0.0020**	0.8209	**0.0053**
GCN-Random (baseline)	0.8246	0.0027	0.8161	0.0058
GCN-Bayes (baseline)	0.8163	0.0042	0.8169	0.0086
GCN-Anneal (baseline)	0.8272	0.0027	0.8182	0.0083
GAT-DE_targetToBest_1_L	0.8398	**0.0017**	0.8271	**0.0058**
GAT-EGA	**0.845**	0.0033	**0.838**	0.0071
GAT-ES_1_PLUS_1	0.8404	0.0020	0.8322	0.0068
GAT-Random (baseline)	0.8366	0.0027	0.8249	0.0091
GAT-Bayes (baseline)	0.8298	0.0026	0.8222	0.0083
GAT-Anneal (baseline)	0.837	0.0018	0.8253	0.0090
Method	Citeseer		Citeseer	
	Average (validation)	Standard deviation (validation)	Average (test)	Standard deviation (test)
GCN-DE_targetToBest_1_L	**0.7414**	**0.0020**	**0.7102**	0.0036
GCN-EGA	0.739	0.0031	0.7101	**0.0029**
GCN-ES_1_PLUS_1	0.738	0.0033	0.7057	0.0090
GCN-Random (baseline)	0.7366	0.0039	0.7045	0.0099
GCN-Bayes (baseline)	0.7267	0.0033	0.6953	0.0069
GCN-Anneal (baseline)	0.739	0.0034	0.7045	0.0074
GAT-DE_targetToBest_1_L	0.746	0.0037	0.7026	0.0138
GAT-EGA	**0.752**	0.0042	**0.7086**	0.0070
GAT-ES_1_PLUS_1	0.7436	0.0041	0.6975	**0.0039**
GAT-Random (baseline)	0.7384	**0.0034**	0.7019	0.0107
GAT-Bayes (baseline)	0.7308	0.0084	0.699	0.0109
GAT-Anneal (baseline)	0.7402	0.0050	0.7026	0.0100

that the HPO with EA perform the better average values and standard deviations of accuracy than the traditional HPO algorithms. Furthermore, under the

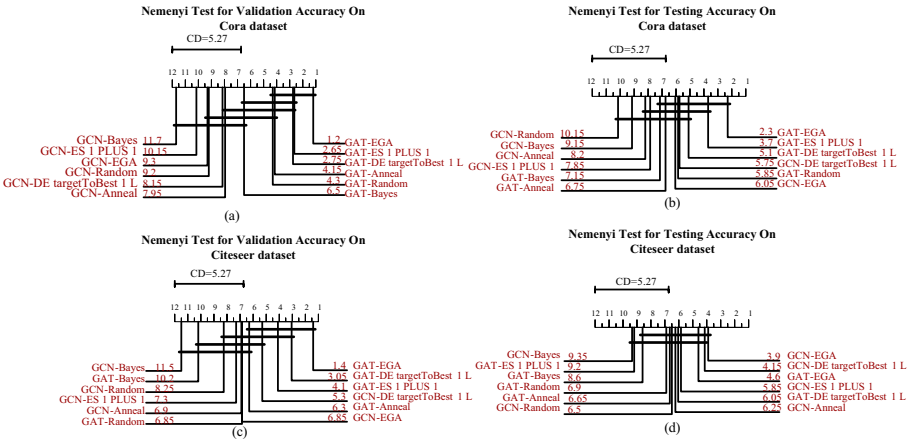

Fig. 4. Nemenyi test results for the validation and testing accuracy on the GCN and GAT tasks. The average rank of each algorithm is marked along the axis (lower ranks to the right). The models on the same horizontal line have a similar prediction performance. The average ranks of the HPO with EA in the validation and testing accuracy outperform those of the traditional HPO methods.

parameter settings and experimental task in this paper, the type of GA owns the excellent performance in the tasks, comparing with DE and ES.

To provide a comprehensive performance comparison between the HPO with EA and traditional HPO, the Friedman and Nemenyi tests [30] were conducted in the experiments, which are widely used to statistically compare different algorithms over multiple datasets. As shown in Fig. 4, the average ranks of the HPO with EA in the validation and testing accuracy outperform those of the traditional HPO methods.

In the Nemenyi tests, it is considered that a significant difference exists if the average ranks of two models differ by at least one critical difference (CD), which is calculated using a 5% significance level. The CD diagrams for the validation and testing accuracy are plotted in Fig. 4, where the average rank of each algorithm is marked along the axis (lower ranks to the right). In Fig. 4, the models on the same horizontal line have similar prediction performance.

3.3 Comparison Among Various Types of EAs

This subsection is to show the performance of various EAs in DE, ES, and GA types, and therefore to provide some options for tasks in AutoGL. They are compared on the GCN and GAT tasks.

Table 6 shows that, for the tested GCN tasks, DE rand bin performs better than other tested DE variants, and ES miu plus lambda performs the best for the tested ES variants. For the GAT tasks, DE best L and ES miu plus lambda performs better than other tested DE and ES variants, respectively. Note that

Table 6. Accuracy performance comparison among the various types of EAs

Method	GCN		GAT	
	Cora	Citeseer	Cora	Citeseer
DE best bin	0.823	0.703	0.843	0.712
DE best L	0.828	0.697	**0.849**	**0.714**
DE rand bin	**0.832**	**0.722**	0.838	0.712
DE rand L	0.823	0.702	0.820	0.709
DE targetToBest bin	0.826	0.703	0.832	0.718
DE targetToBest L	0.826	0.705	0.839	0.713
DE currentToBest bin	0.826	0.705	0.838	0.729
DE currentToBest L	0.827	0.709	0.823	0.706
DE currentToRand	0.827	0.703	0.825	0.720
ES plus	0.814	0.704	0.823	0.706
ES miu plus lambda	**0.816**	**0.719**	**0.828**	**0.721**
EGA	**0.825**	0.721	0.845	0.709
SEGA	0.817	0.717	**0.847**	0.710
SGA	0.818	0.710	0.838	**0.714**
studGA	0.810	**0.722**	0.832	0.710

these are under the specific parameter settings and experimental tasks in this paper.

In addition, the effect of EA parameters on the tested tasks is shown in Fig. 5. The adjusted parameters are population size, crossover rate, and mutation rate.

3.4 Findings

To summary, several findings are concluded as follows. It should be pointed out that the analysis and the findings are under the experimental settings in this paper. The comparison results may be different in different experimental settings or tasks.

1. When the classification accuracy on the verification set is considered as the target to be optimized, most HPO methods using EAs perform better than the optimal method among traditional HPO methods.
2. When considering the type of EAs in the HPO of GNNs for good performance, GA is a good choice based on the experiments under the specific parameter setting in this paper.
3. According to our experiments, DE rand bin, DE best L, and ES miu plus lambda might be good options for optimizing the GCN and GAT. And the experimental results of genetic algorithm are relatively stable for the tested cases.

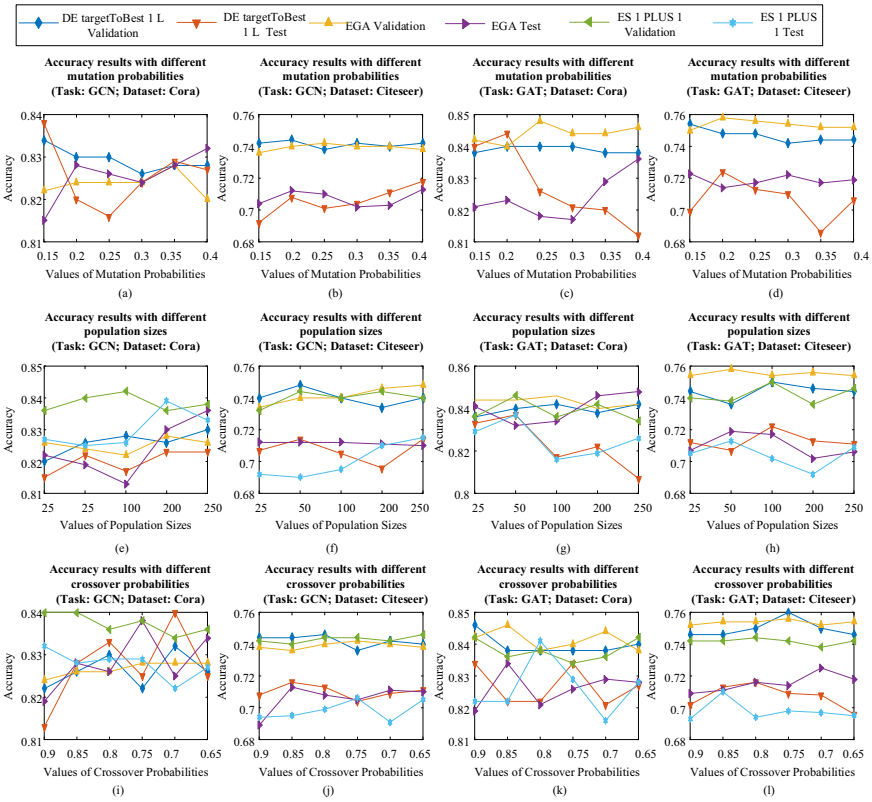

Fig. 5. Influence of the parameters of evolutionary algorithm, including the mutation probability, the population size, and the crossover probability. Three algorithms, i.e., DE targetToBest 1 L, EGA, and ES 1 Plus 1, are conducted on the GCN and GAT tasks on the validation and testing data of Cora and Citeseer datasets.

4 Conclusion

In this study, we aim to demonstrate the validity of EAs in the HPO work of automatic graph neural network learning. We have performed many experiments to test the performance differences of different EAs. The experiments are conducted on the Cora and Citeseer datasets. The experimental results show that the EAs could be an effective alternative to the hyperparameter optimization of GNN, which could provide a reference for later researchers interested in this field.

References

1. Elshawi, R., Maher, M., Sakr, S.: Automated machine learning: State-of-the-art and open challenges, pp. 1–23 (2019). arXiv preprint arXiv:1906.02287

2. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning. TSS-CML, Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5
3. He, X., Zhao, K., Chu, X.: Automl: a survey of the state-of-the-art. Knowl.-Based Syst. **212**(106622), 1–27 (2021)
4. Guan, C., et al.: Autogl: a library for automated graph learning, pp. 1–8 (2021). arXiv preprint arXiv:2104.04987
5. Liu, Y., Zeng, K., Wang, H., Song, X., Zhou, B.: Content matters: a GNN-based model combined with text semantics for social network cascade prediction. In: Karlapalem, K. (ed.) PAKDD 2021. LNCS (LNAI), vol. 12712, pp. 728–740. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75762-5_57
6. Fan, W., et al.: Graph neural networks for social recommendation. In: Proceedings of The World Wide Web Conference, pp. 417–426. ACM (2019)
7. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2020)
8. Liu, M., Gao, H., Ji, S.: Towards deeper graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 338–348. ACM (2020)
9. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Proceedings of 2005 IEEE International Joint Conference on Neural Networks, 2005, vol. 2, pp. 729–734. IEEE (2005)
10. Zhou, J., et al.: Graph neural networks: a review of methods and applications. AI Open **1**(1), 57–81 (2020)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks, pp. 1–14 (2016). arXiv preprint arXiv:1609.02907
12. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks, pp. 1–12 (2017). arXiv preprint arXiv:1710.10903
13. Bu, C., Luo, W., Yue, L.: Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. IEEE Trans. Evol. Comput **21**(1), 14–33 (2017)
14. Dang, D., Jansen, T., Lehre, P.K.: Populations can be essential in tracking dynamic optima. Algorithmica **78**(2), 660–680 (2017)
15. Zhou, Z., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer, Heidelberg (2019). https://doi.org/10.1007/978-981-13-5956-9
16. Eiben, A.E., Smith, J.: From evolutionary computation to the evolution of things. Nature **521**(7553), 476–482 (2015)
17. Bu, C., Luo, W., Zhu, T., Yue, L.: Solving online dynamic time-linkage problems under unreliable prediction. Appl. Soft Comput. **56**, 702–716 (2017)
18. Zhu, T., Luo, W., Bu, C., Yue, L.: Accelerate population-based stochastic search algorithms with memory for optima tracking on dynamic power systems. IEEE Trans. Power Syst. **25**, 268–277 (2015)
19. Yi, R., Luo, W., Bu, C., Lin, X.: A hybrid genetic algorithm for vehicle routing problems with dynamic requests. In: 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, 27 November–1 December 2017, pp. 1–8. IEEE (2017)
20. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 2902–2911. PMLR (2017)

21. Bu, C., Luo, W., Zhu, T., Yi, R., Yang, B.: Species and memory enhanced differential evolution for optimal power flow under double-sided uncertainties. IEEE Trans. Sustain. Comput. **5**(3), 403–415 (2020)
22. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of The 33rd International Conference on Machine Learning, pp. 40–48. PMLR (2016)
23. Storn, R.: On the usage of differential evolution for function optimization. In: Proceedings of North American Fuzzy Information Processing, pp. 519–523. IEEE (1996)
24. Opara, K.R., Arabas, J.: Differential evolution: a survey of theoretical analyses. Swarm Evol. Comput. **44**(1), 546–558 (2019)
25. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. NCS, Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-31306-0
26. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2010)
27. Beyer, H.G., Schwefel, H.P.: Evolution strategies-a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002)
28. Holland, J.H., et al.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press, Cambridge (1992)
29. Khatib, W., Fleming, P.J.: The stud GA: a mini revolution? In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 683–691. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0056910
30. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(1), 1–30 (2006)