Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 12, No. 1, 2024, 115-126.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Automatic Configuration of Federated Learning Client in Graph Classification using Genetic Algorithms

Mohammad Rezaei and Mohsen Rezvani* and Morteza Zahedi

Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran.

Article Info

Article History:
Received 05 November 2023
Revised 30 November 2023
Accepted 02 February 2024

DOI:10.22044/jadm.2024.13688.2487

Keywords:
Federated Learning, Model Agnostic Meta-Learning (MAML), Federated Learning on Graph

**Corresponding author: mrezvani@shahroodut.ac.ir (M. Rezvani).*

Abstract

With the increasing interconnectedness of communications and social networks, graph-based learning techniques offer valuable information extraction from data. Traditional centralized learning methods faced challenges, including data privacy violations and costly maintenance in a centralized environment. To address these, decentralized learning approaches like Federated Learning have emerged. This study explores the significant attention Federated Learning has gained in graph classification and investigates how Model Agnostic Meta-Learning (MAML) can improve its performance, especially concerning non-IID (Non-Independent Identically Distributed) data distributions.

In real-world scenarios, deploying Federated Learning poses challenges, particularly in tuning client parameters and structures due to data isolation and diversity. To address this issue, this study proposes an innovative approach using Genetic Algorithms (GA) for automatic tuning of structures and parameters. By integrating GA with MAML-based clients in Federated Learning, various aspects, such as graph classification structure, learning rate, and optimization function type, can be automatically adjusted. This novel approach yields improved accuracy in decentralized learning at both the client and server levels.

1. Introduction

Tremendous advances in communication technologies have led to an unprecedented surge in data generation. In the realm of graphs, social network data [1], financial transactions [2, 3], and biological networks [4] stand out as vital sources of information that continue to grow exponentially. Graph learning endeavors to extract valuable insights from such data, employing models like graph regularization [5, 6], graph embedding [7], and graph neural networks [8]. Notably, research in community detection [9-11], personalized recommendation [12, 13], and fraud detection [3, 14-16] underscores the paramount importance of investigating graph-based datasets.

Previous studies have mainly focused on centralized learning models, where training and data storage occur in a centralized manner. However, in real-world scenarios, data is often distributed among separate entities, such as organizations or industries [17]. For example, each bank has its own customer information, network, and transaction history. One crucial model that could be created with the collaboration of banks is the evaluation, detection, and creation of a blacklist of customers. To achieve this, banks would need to share their information in a centralized environment and use graph learning to create the desired model. However, due to competition and data security concerns, this task becomes challenging, making it difficult to create a high-quality model by combining data from various entities [18].

Federated learning is a decentralized approach to training data that aims to create a shared model by utilizing data and resources from different devices [8, 19]. The main idea of federated learning is to

aggregate updates sent by local model training on each device to create a shared model while preserving data privacy and security [20, 21]. Figure 1 provides a general illustration of federated learning applied to a banking system. In recent years, various articles have been published on federated learning for graph classification, showing promising results and addressing some of the challenges of centralized learning [22-25]. Key advantages of using federated learning for graph classification include preserving data privacy and utilizing external resources for training (client training resources).

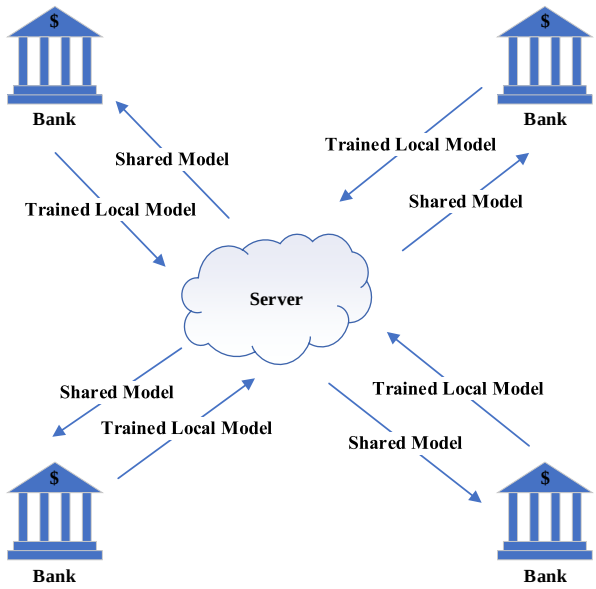

Figure 1. Federated Learning in a Banking System

While federated learning offers significant advantages, it also comes with challenges. One of the main challenges is dealing with the non-IID (non-independent identically distributed) linear distribution of data [20, 21]. The use of model-agnostic meta-learning (MAML) has been proposed to reduce the impact of non-IID data distribution and improve the performance of federated learning [20, 21]. However, other challenges remain when applying federated learning, such as dealing with a large number of federated clients in the real world. Many existing methods require manual settings and fixed client structures during the federated learning algorithm, which can be costly in real-world scenarios due to human errors and parameter tuning [26-29]. Furthermore, using the same settings and structure for all clients may not yield the best performance for each client in the real world.

In this paper, we introduce an innovative federated learning algorithm tailored for graph classification, harnessing the combined power of model-agnostic meta-learning and genetic algorithms. Our proposed algorithm not only automates the intricate process of client tuning but also deftly addresses the myriad challenges that have been elucidated earlier. Model-agnostic meta-learning is strategically employed to alleviate the detrimental effects of non-IID data distribution, enhancing the model's adaptability. The genetic algorithms, on the other hand, play a pivotal role in seamlessly adjusting client-specific parameters. The overarching goal is to realize enhanced performance, not only at the individual client level but also across the aggregate model.

In essence, the crux of our approach resides in affording each client the capability to iteratively refine its structural configuration and pertinent parameters through the employment of genetic algorithms. By leveraging knowledge accrued in each training cycle and capitalizing on its transferability to subsequent iterations, we anticipate a notable enhancement in client accuracy. The initial experiment lays bare the potential inherent in preserving program environment parameters and facilitating knowledge transfer, ultimately resulting in improved client accuracy. This foundation then supports the core assertion of our method's superiority, substantiated through a series of meticulously designed experiments.

Our experiments meticulously demonstrate the tangible benefits that stem from the marriage of model-agnostic meta-learning and genetic algorithms. The proposed method exhibits an encouraging capacity to uplift client performance through the dynamic evolution of their individual architectures and parameters. This dynamic approach not only imbues clients with improved accuracy but also underscores the algorithm's efficacy in generating a more potent final model. The ensuing sections detail these experiments, offering comprehensive insights into our method's methodology, its experimental setups, and the key outcomes that validate its effectiveness.

We summarize our contributions as follows:

- Reducing the impact of human errors in client parameter tuning and structure adjustment
- Automating the determination of client structures and parameters
- Automatically creating different parameters and structures for each client
- Enabling client tuning without requiring prior access and knowledge of data nature
- Improving the performance (in terms of accuracy) of both client and server models compared to previous models

The paper proceeds by reviewing existing works related to genetic algorithms and federated learning in the context of graph classification in Section 2. Section 3 presents the proposed method and the structure of the genetic algorithm used. Finally, the paper presents and analyzes experimental results to evaluate the performance of the proposed method in Section 4, which is followed by the concluding remarks.

2. Review of Previous Work

In this section, we provide an overview of relevant prior research that employs the genetic algorithm for automatic structure generation. Our focus is on highlighting the key differences between these previous works and our proposed approach, showcasing the novel aspects of our contribution.

We begin by examining research that employs the genetic algorithm for the automatic design of convolutional neural network structures in image classification [26]. Conventional methods often require expert intervention and extensive experimentation to determine the optimal network architecture. Existing algorithms like GoogleNet, ResNet, and DenseNet have been developed, each with distinct structures. Addressing the challenge of selecting an appropriate network architecture involving factors like depth, layer parameters, and interconnections, Suganuma et al. demonstrated the effectiveness of their genetic algorithm [26]. By comparing their approach to others, they not only generated and adjusted network structures but also achieved notable gains in accuracy and speed. Notably, their algorithm excelled at handling complex structures within image processing. An essential outcome of this study was uncovering the correlation between convolutional neural network structures and data volume.

Similarly, Gibb et al. utilized the genetic algorithm to optimize convolutional neural network structures for crack detection in concrete [27]. The intrinsic challenges of crack detection encompass issues such as varying lighting conditions, background noise, and crack morphology. While convolutional neural networks have proven effective, manual tuning of their intricate parameters can be time-intensive. To mitigate this, Spencer et al. introduced an algorithm for automatic network structure adjustment, outperforming conventional hand-tuned approaches.

Another contribution by Sun et al. addressed the influence of convolutional neural network structures on performance [28]. Their approach leveraged a genetic algorithm to automate network structure design. By contrasting their method with alternative strategies, Sun et al. showcased improved classification accuracy, parameter values, and computational efficiency.

Another application of genetic algorithms for parameter automation can be observed in the work of Falahiazar et al., where they employ a genetic algorithm to determine the parameters of the DBSCAN algorithm[30].

Transitioning to the foundation of our proposed method—federated learning—we evaluate work that lays the groundwork for introducing the genetic algorithm to automatically optimize client parameters within federated learning settings.

The 2020 article introduced the first application of model-agnostic meta-learning in graph classification within the framework of federated learning, which is known as GraphFL[21]. GraphFL successfully tackled significant challenges often encountered in federated learning scenarios, including addressing non-IID linear data distribution and improving performance when dealing with unseen labels during training. By harnessing the power of model-agnostic meta-learning, the GraphFL article introduced an innovative solution to graph classification within the federated learning context, thereby offering a fresh perspective on the optimization of parameters.

To summarize, our review of previous work showcases the evolution of the genetic algorithm's application across different domains, from optimizing convolutional neural network structures to enhancing federated learning settings. Our proposed approach not only builds upon these foundations but also introduces novel strategies for automatic structure generation and parameter optimization.

3. Proposed Method

Federated learning consists of two main components: the server and the clients, which are referred to as the global or shared model for the server and the local model for the clients. In the context of our study, we adopt specific terminology to distinguish between operations conducted at the server level and those executed at the client level. We utilize the term "stage" to denote various operations performed on the server, encompassing overarching processes that guide the federated learning framework. Conversely, at the client level, we employ the term "step" to signify discrete operations, encompassing actions that transpire within each individual client's local environment. This clear distinction is instrumental in elucidating the multifaceted dynamics of our proposed methodology, facilitating a comprehensive

understanding of the interplay between different layers of the federated learning process.
Figure 2 provides an overview of the proposed federated learning framework in both the server and each client. It follows the subsequent steps and stages:

1) In the first stage, the initial global model is created using initial settings on the server.
2) In the second stage, participating clients are selected. The selection is done randomly, and all clients have an equal chance of being chosen.
3) After determining the clients, their training stage begins. In this part, the global model is sent to the participating clients. Each client proceeds through the following steps: In the first step, each client runs the genetic algorithm using the global model and its local data. The output of this operation is a new model for classification and the required parameters for model-agnostic meta-learning, which will be thoroughly examined in the genetic section. In the second step, client training is performed using the model-agnostic meta-learning algorithm. As mentioned in the previous step, the parameters and structure used for training in this step are derived from the output of the genetic algorithm. The last step involves evaluating the trained model and reporting its accuracy to the clients.
4) The next stage is the update of the global model. After completing the training of each client, the newly trained model is sent back to the server. The server waits until all clients finish their training (concurrently) and then proceeds to aggregate and update its model. The aggregation method used for combining models is FedAvg [31, 32], which is also employed for training the next round of clients. The next training iteration starts from step two, and these steps are repeated until a termination condition is met.

It is worth mentioning that this article proposes three ways to apply the genetic algorithm, which are discussed in Section 3.2. The general nature of the proposed federated learning is described above, and in Section 3.2, the necessary modifications to the federated learning structure are presented based on the intended application of the genetic algorithm.

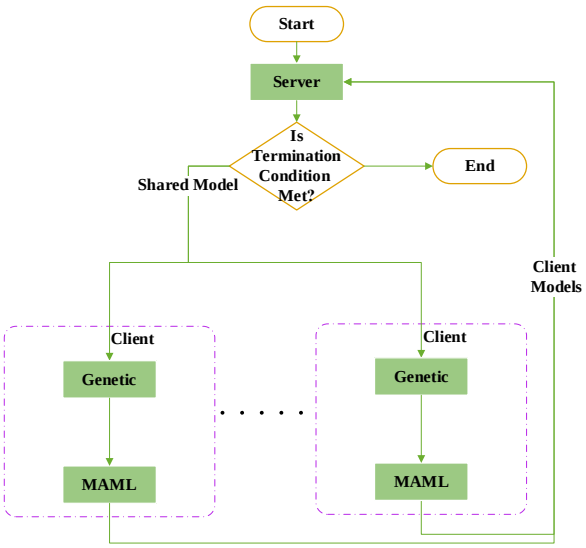

Figure 2. Proposed federated learning algorithm overview for both server and clients.

3.1 Genetic Algorithm

In this section, the genetic algorithm used in the proposed method is discussed. The key and innovative part of the proposed approach is the utilization of the genetic algorithm. Hereafter, the term "step" is used to describe the algorithm stages, which are highlighted in bold in the text. The genetic algorithm follows the steps outlined below to optimize the structure and parameters.

1) In the initial step, the initial population is created. Each chromosome in the population represents parameters and structures that can be used for model-agnostic meta-learning and classification. In the first round of training the global model, one of the chromosomes in the genetic algorithm population represents the environment parameters of execution. In the subsequent rounds, the population and chromosomes from the previous training are utilized.
2) In the second step, the evaluation function is called to assess the accuracy and obtain the new model for the chromosome. In the evaluation function, first, the new model is created using the chromosome information and the global model. Then, the local data (client local data) is used to evaluate the chromosome's value through the model-agnostic meta-learning algorithm. It is worth mentioning that the model-agnostic meta-learning algorithm and graph-based classification used in the evaluation function are new models created at the beginning of the function using chromosome information. If the algorithm identifies the chromosome as a solution to the problem, this model will be used

as the initial training model for the client instead of the global model used in the GraphFL algorithm.

3) In the subsequent steps, genetic algorithm operators are used to create a new population. In the third step, chromosomes with the highest value are preserved based on a specific proportion, which can be considered the reason for creating chromosomes with environment parameters in the initial step. If the environment parameters achieve the best client training state, adding them to the initial chromosome preserves the best state and assists in creating chromosomes with higher values in the subsequent steps. Also, if the environmental parameters do not have a high value, they will be removed from future generations, and chromosomes with higher values will take their place.
4) In the fourth step, chromosomes from the population are combined pairwise to create new chromosomes for the next generation. Some of the chromosomes used in the combination are the ones with the highest values selected in the previous step.
5) The fifth step involves applying the mutation operator, which occurs randomly after the combination. Some chromosomes may undergo mutation, while others may not use the mutation operator.

Steps two to four are repeated until the termination condition is met. At the end of the genetic algorithm, the best model and parameters obtained are returned. Figure 3 represents the structure of the algorithm.

An important point in the genetic algorithm is the selection operator. This operator is responsible for selecting chromosomes for the next generation and their combination. In the proposed genetic algorithm selection operator, a portion of chromosomes with the best scores is preserved for the next generation based on a specific proportion. Then, another portion of chromosomes is randomly selected. This process is based on a specified ratio. Preserving a portion of chromosomes with the highest scores leads to their retention in the subsequent generations. Additionally, since they are used in the combination operator, they also help create better chromosomes.

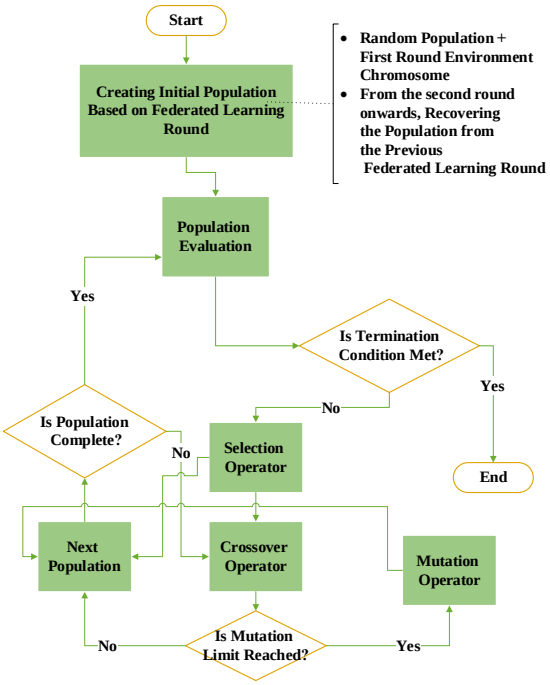

Figure 3. Structure of the proposed genetic algorithm.

3.2. Evaluation Function

3.2.1. Accuracy-based Evaluation

One of the most important tasks of this evaluation function is to assess the chromosomes. The evaluation function used in the proposed genetic algorithm not only validates the chromosomes but also generates new models. The chromosome and the global model are considered the most significant inputs to this function. This function creates a new model using the chromosome and the global model in the initial stage. Then, it proceeds to train this model using its local training data. In the next step, the trained model is evaluated with local test data, and the accuracy and the learned model are the outputs of this function, as depicted in Figure 4, which presents an overview of this evaluation function.

3.2.2. Accuracy and Time-based Evaluation

Most parts of this method are similar to the previous evaluation function. In this method, after creating the model with chromosome information, a data point called "start time" is generated at the beginning of the model training stage, and another data point called "end time" is created at the end of the testing stage. The training time is obtained by calculating the difference between the end time and the start time. This training time, along with the accuracy computed in the testing stage, is used to evaluate the value of each chromosome.

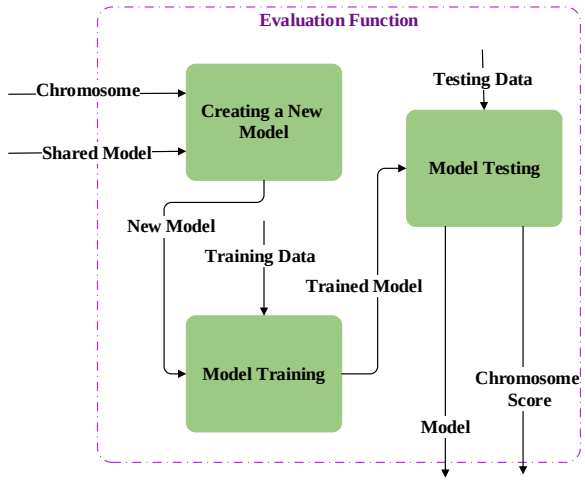

Figure 4. General schematic of the proposed evaluation function

3.3. Using the Genetic Algorithm

In this section, three proposed methods for utilizing the genetic algorithm are introduced, each with its own advantages and specific considerations to address the time and resource challenges posed by the proposed algorithm. The following subsections present these methods:

3.3.1. Continuous Method

This approach employs a continuous genetic algorithm within the clients to continually optimize their model structures. A challenge associated with this method is the significant time investment required for client training. The process is visually depicted in Figure 5.

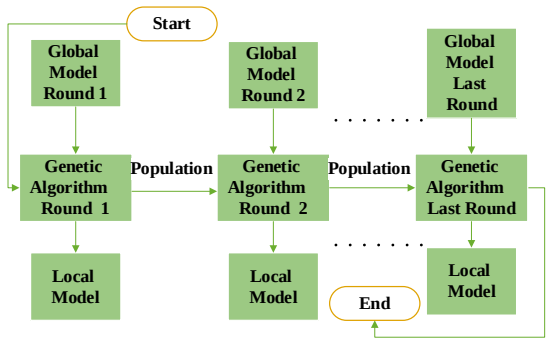

Figure 5. Continuous Method: Clients iteratively refine local models using a genetic algorithm and prior knowledge.

3.3.2. Parallel Method

In this method, the genetic algorithm's execution is decoupled from the client training process. Upon determining a new structure, the algorithm integrates the shared model weights for the current round. Subsequently, the genetic algorithm is executed in a separate process. Notably, this technique demands higher processing resources than other methods due to its use of parallel programming within the clients.

3.3.3. Threshold Method

The Threshold Method categorizes clients into two lists based on a predefined threshold set by the server. Clients with metrics surpassing the threshold are placed in the trainable list, while others requiring structural changes reside in a separate list. In the initial round, all clients are positioned in the trainable list, and the server continuously selects from this list. After client training, a metric threshold is assessed; if it falls below the defined threshold, the genetic algorithm is employed for structural determination. Subsequently, the server relocates the client to the list of clients necessitating structural modifications. Upon determining the new structure, the client transitions back to the trainable list, employing the new structure in training when selected.

3.4. Time Complexity

In this section, we analyze the time complexity of the Continuous method compared to the GraphFL method. Assuming that the GraphFL method has a time complexity denoted as "en", we anticipate the proposed algorithm to exhibit the following time complexity. The algorithm's complexity is contingent on the following parameters: "e", representing the number of epochs on the server; "n", indicating the training time of the last client; "m", denoting the number of rounds in the genetic algorithm; "c", representing the number of chromosomes; "A", a constant value (pertaining to genetic operations excluding the evaluation function); and "B", another constant value (related to new operations in the client, such as the evaluation stage).

$$e(m(cn + A) + n + B)$$

It should be noted that in both algorithms, owing to the concurrent implementation of training, the time complexity of the clients equals that of the slowest client.

3.5. Conclusion

The proposed method employs a genetic algorithm with a Mixed-Type Representation. Chromosome selection is performed through a combination of Threshold-Based Selection with Retention and Random Selection. The crossover operation utilizes Uniform Crossover, while the mutation process employs Random Mutation. In the context of selection, Retention is applied to ensure the preservation of the best models in each round.

Notably, in each subsequent training round, the chromosomes from the last round are retained as the first chromosomes, contributing to the continuity of the evolutionary process. This comprehensive approach enhances the algorithm's ability to explore diverse solution spaces and maintain the efficacy of the evolved models throughout successive iterations.

4. Evaluation Results

The proposed method's structure and algorithms were discussed in the previous section. In this section, the proposed method is compared with the GraphFL algorithm [21] through various experiments. One of the closest approaches to the proposed method is the GraphFL algorithm, where both the proposed method and GraphFL use MAML to improve the performance of federated learning. The implementation and experiments with the GraphFL algorithm were carried out in a consistent environment with the proposed method.

4.1. Settings

The experiments in this article were conducted using two machines without utilizing GPU. Experiments involving fewer than six clients were performed on a machine with 5 CPU cores, while other experiments were conducted on a machine with more than 20 cores. Both algorithms were implemented using the PyTorch framework and the Higher and PyG (PyTorch-Geometric) libraries. Python was the main programming language for these implementations. PyTorch served as the primary framework, Higher was utilized for implementing MAML for model-agnostic meta-learning, and PyG was a powerful library for graph-related tasks. The training of clients was parallelized to increase the experiment's response speed. Another parameter of the proposed algorithm was the use of a continuous genetic method and precision as the evaluation metric to assess the chromosome's value in the evaluation function.

The adoption of the SGC algorithm in our study is substantiated by its remarkable performance in graph classification within the federated learning framework, as evidenced by the GraphFL article. Given its demonstrated efficacy, we conducted our experiments leveraging the SGC algorithm to ensure a robust evaluation. The choice of performance metric aligns with that used in the GraphFL article, where Accuracy served as the benchmark. Consequently, our proposed algorithm underwent implementation under comparable conditions to GraphFL, facilitating a meaningful and equitable evaluation based on this common metric.

Genetic parameters for the experiments are configured with a retention rate of 0.4, a random selection rate of 0.1, a crossover rate of 0.5, and a mutation rate of 0.2. The experiments involve 10 iterations with a population size of 20.

The high retention rate is implemented to safeguard the most valuable chromosomes and prevent excessive similarity among them. Furthermore, 0.1 chromosomes are randomly chosen from the remaining population to introduce variability into the selection process.

Upon the completion of iterations, the best model will be chosen to train the client model.

All experiments were conducted using the CORA dataset [33, 34], which includes 2708 scientific publications classified into seven classes. The citation network comprises 5429 edges. Each publication in the dataset is described by a binary word vector representing the presence or absence of specific words in the vocabulary, consisting of 1433 unique words.

The figures presented in the experiments pertain to the evaluation phase conducted on the clients. The tables corresponding to these experiments, which are presented below, are associated with the testing phase carried out on the server.

4.2. The Effect of Genetic Algorithm on Clients

In this section, the results of comparing the proposed method with GraphFL and the impact of different conditions on the genetic algorithm are presented. In these experiments, the training, evaluation, and test data were randomly generated for each client, with a maximum of 6 clients used. Due to the concurrent execution of both algorithms, GraphFL benefits from a similar environment to the proposed algorithm.

Figure 6a shows the experimental results where the initial population of the genetic algorithm is randomly generated, and the genetic algorithm is not used to create an optimal structure for the classification algorithms. As observed, the proposed algorithm performs poorly in some clients, but due to its significantly better performance in other clients, it leads to overall improvement during the testing phase (a 2% improvement in the testing phase was observed).

Among other experiments conducted in this section, the addition of environmental execution parameters (number of workers, number of support sets, number of target sets, learning rate, optimization function type, and other parameters) to the initial population of the genetic algorithm in the initial rounds of each client was tested. The

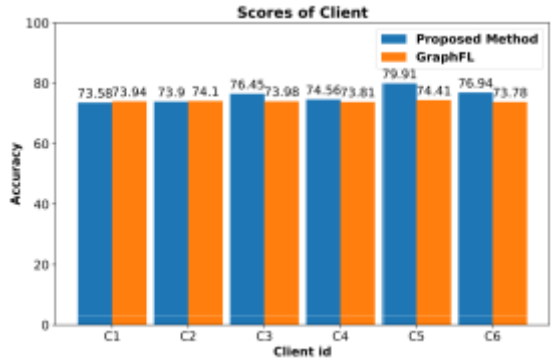

Figure 6a. Evaluating Client Accuracy with Randomly Generated Initial Genetic Algorithm Population.

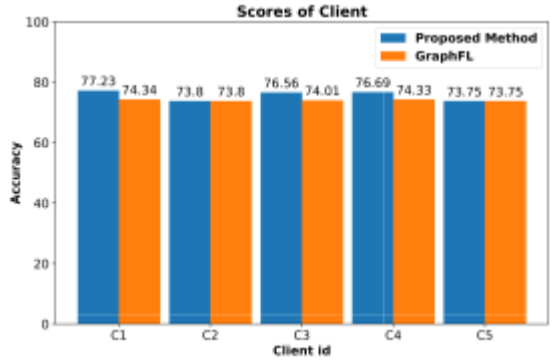

Figure 6b. Evaluating Client Accuracy with Environmental Parameters in the Initial Genetic Algorithm Population.

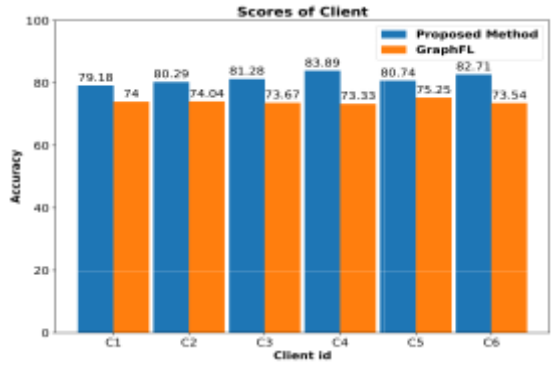

Figure 6c. Assessing Client Accuracy with Environmental Parameters and Graph Structure in the Genetic Algorithm.

As shown in Table 1. Environment and Results from Test Phase in Experiment Examining Genetic Algorithm Impact on Clients., modifying clients, especially when the classification structure is modified, leads to a considerable improvement in federated learning performance. In subsequent experiments, the genetic algorithm will be used with both environmental parameters and the ability to modify the classification structure.

results of this experiment are presented in Figure 6b. These parameters can also be considered as GraphFL parameters. This experiment showed that the addition of parameters prevents poor client performance. Due to the low number of clients and very few rounds, a 0.8% improvement was observed during the testing phase (in this experiment, the genetic algorithm was not responsible for changing the classification structure).

After adding the classification structure change to the genetic algorithm, a significant improvement was observed in clients, as shown in Figure 6c. Besides, the algorithm demonstrated excellent performance in the testing phase. A nearly 10% improvement in the testing phase indicates the importance of classification structure changes in the genetic algorithm, as detailed in Table 1.

Table 1. Environment and Results from Test Phase in Experiment Examining Genetic Algorithm Impact on Clients.

Experiment ID	Number of Clients	Data Distribution Method	GraphFL Accuracy in the Test Phase	Proposed Method Accuracy in the Test Phase
Figure 6a	6	Random	73.48	76.24
Figure 6b	5	Random	74.01	74.80
Figure 6c	6	Random	73.91	83.26

4.3. Impact of the Number of Participating Clients in Training

This experiment aims to investigate the algorithm's performance with non-uniform data (non-uniform data distribution) and observe the performance of clients under such conditions. In this section, 50 clients are formed for the experiments, and the data distribution among the clients is randomized. The nature of clients, in terms of data and other settings, is the same for both the proposed method and GraphFL at the beginning of the experiments. Two experiments were conducted in this section, using 10% and 20% of the clients to participate in the model training.

Figure 7a and Figure 7b present the results obtained from calculating the accuracy of the experiments conducted in this section. In the first experiment, 5 clients, or 10% of the clients, were used in the training. As observed, the algorithm's performance is superior to GraphFL on all clients. In the next experiment, 10 clients, or 20% of the clients, participated in the training as depicted in Figure 7b. As shown in the images, with an

increase in the number of clients, due to the reduction in the non-uniform data distribution problem, the performance of clients has improved. Additionally, the overall performance of clients has improved as well. Table 2 shows the results obtained from the testing phase of the experiments.

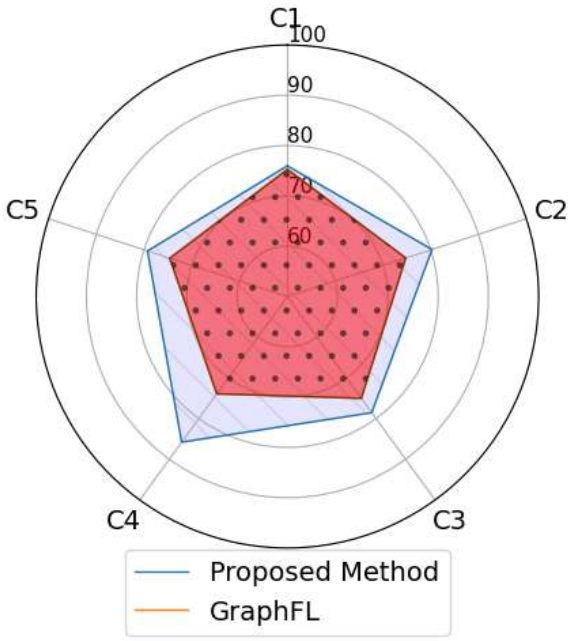

Figure 7a. Clients Accuracy in the Experiment with 10% of Clients Participating in Training.

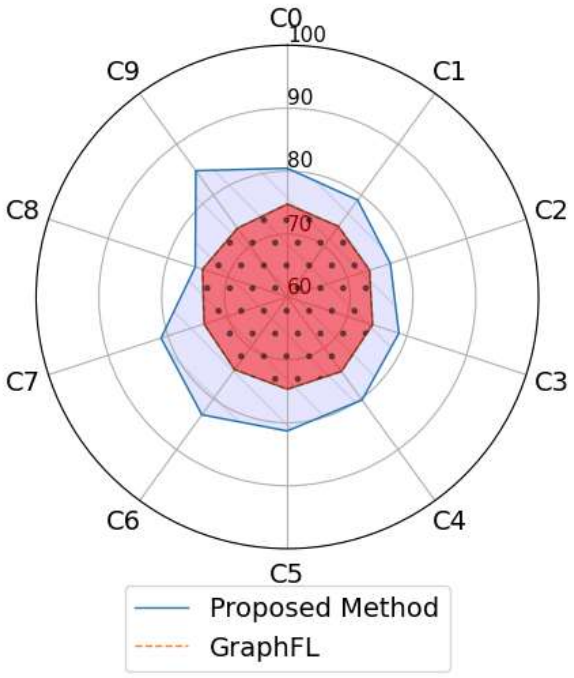

Figure 7b. Clients Accuracy in the Experiment with 20% of Clients Participating in Training.

Table 2. Results from Test Phase in Experiment with Varying Number of Participating Clients in Training.

Experiment ID	Percentage of participating clients	Accuracy of GraphFL in the Test Phase	Accuracy of the Proposed Method in the Test Phase
Figure 7a	10	74.31	83.24
Figure 7b	20	74.09	83.26

4.4. Impact of Data Overlapping

The exploration now transitions to the investigation of data overlapping, a critical facet that merits thorough analysis within the context of this study. An experiment involving a 20% node overlapping scenario was methodically executed, yielding outcomes showcased in Figure 8. To present a comprehensive overview, Table 3 succinctly details the outcomes from the testing phase, specifically on the proposed method's performance. The results undeniably highlight the efficacy of the proposed approach in this experiment, demonstrating a satisfactory level of performance with noteworthy accuracy during testing and elevated accuracy amongst the individual clients.

Table 3. Results from Test Phase in Experiment with Varying Client Participation in Training.

Experiment ID	Number of Clients	Accuracy of GraphFL in the Test Phase	Accuracy of the Proposed Method in the Test Phase
Figure 8	5	81.03	83.97

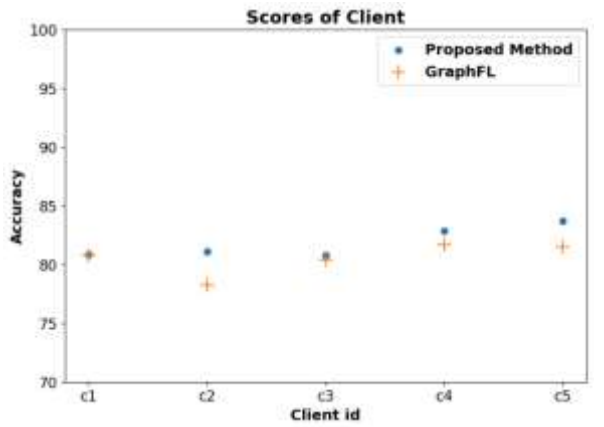

Figure 8. Client Accuracy in the Experiment of the Impact of Data Overlapping.

4.5. Impact of Nodes with New Labels on Testing

The following investigation delves into a comprehensive experiment to assess the performance of federated learning under controlled label distribution and imposed limitations on labels during the training phase. Through a systematic comparison between the proposed method and GraphFL, the experiment substantiates the superior performance of the proposed approach. This empirical endeavor mirrors a scenario that is plausible in real-world contexts, where occurrences such as label expansion and the introduction of new labels are commonplace.

The experiment in this section involved 5 labels during the training phase. As shown in Figure 9. Accuracy of Clients in the Test of the Impact of Nodes with New Labels, the clients demonstrated good performance compared to the GraphFL method.

Table 4 reports the results obtained from the testing phase for the proposed method and GraphFL. As observed, the accuracy of the proposed method in the testing phase is higher than GraphFL. The slight decrease in accuracy during the testing phase can be attributed to the absence of some labels during the training phase.

Table 4. Results from Test Phase: Experiment on Impact of Nodes with New Labels on Testing.

Experiment ID	Number of Clients	Accuracy of GraphFL in the Test Phase	Accuracy of the Proposed Method in the Test Phase
Figure 9	5	77.12	78.78

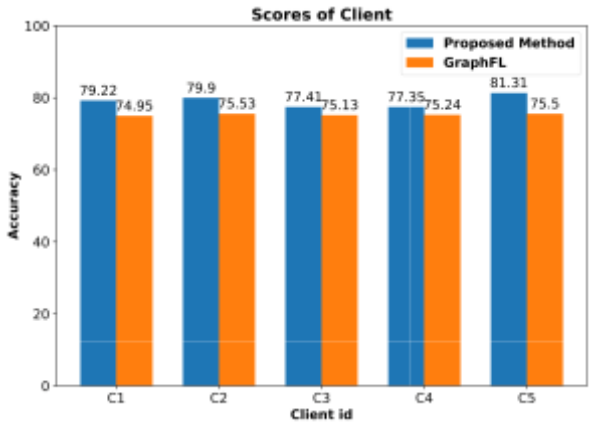

Figure 9. Accuracy of Clients in the Test of the Impact of Nodes with New Labels.

4.6. Investigation of the Impact of Labeled Nodes

This experiment was conducted with controlled label distribution among clients to examine the effect of non-uniform data distribution. The experiment utilized a total of 20 clients. Both the proposed algorithm and GraphFL used the same dataset for training, testing, and throughout all stages of the experiment. The data distribution in this experiment is controlled by dividing nodes among clients. Due to the large number of clients, each experiment was divided into two charts.

The experiments were conducted with 14, 16, and 17 labeled nodes for each class. As demonstrated in the experiments, the client accuracy showed significant improvement due to the structure modification in federated learning and graph-based classification. Figure 10. Client Accuracy in the Node Labeling Experiment with 14 Nodesillustrates the results obtained from the accuracy calculation in the experiment with 14 labeled nodes.

Table 5 presents the results obtained from the testing phase in the experiment on the impact of labeled nodes.

Table 5. Results Obtained from the Test Phase in the Impact Analysis of Labeled Nodes.

Labeled Nodes	Accuracy of GraphFL in the Test	Accuracy of the Proposed Method in the Test
14	73.25	80.54
16	76.32	81.02
17	74.75	81.50

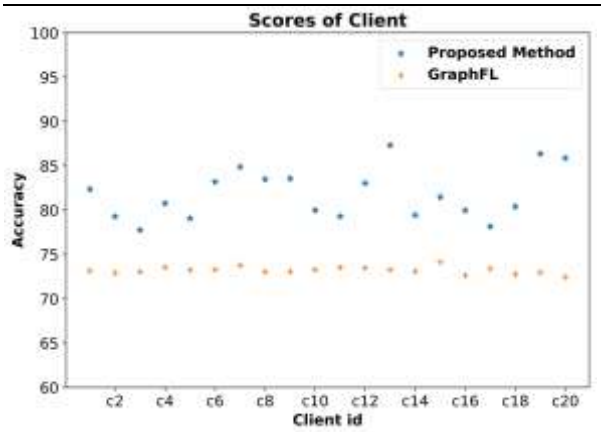

Figure 10. Client Accuracy in the Node Labeling Experiment with 14 Nodes.

5. Conclusion

In this paper, a method for automating federated learning clients using a genetic algorithm in the context of graph-based classification was proposed. As demonstrated in the experiments, the

proposed method with the ability to autonomously adjust clients leads to an improvement in federated learning performance. Another advantage of using the proposed method is the individual structuring of clients, where each client can have a different structure based on its dataset and environment compared to other clients. Further research directions based on the proposed algorithm are discussed below:

1) While the focus of this paper is on graph-based classification, the genetic-inspired approach is drawn from image processing literature. Therefore, image processing can be considered as another research area based on the proposed algorithm.

2) Addressing the speed of response from clients stands out as a significant challenge in the proposed algorithm. Acknowledging this limitation, we recognize the need for future research to delve into improving both the speed and overall structure of our method. Furthermore, it is imperative to undertake a comprehensive analysis of the computational complexity of our proposed algorithm and compare it with GraphFL, as suggested, to provide a more holistic understanding of its performance characteristics about existing methodologies.

References

[1] J. Scott, What is social network analysis? Bloomsbury Academic, 2012. [E-book] Avilable: oapen

[2] A. Khazane *et al.*, "DeepTrax: Embedding graphs of financial transactions," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019: IEEE, pp. 126-133.

[3] R. Li, Z. Liu, Y. Ma, D. Yang, and S. Sun, "Internet financial fraud detection based on graph learning," *IEEE Transactions on Computational Social Systems*, 2022.

[4] G. A. Pavlopoulos *et al.*, "Using graph theory to analyze biological networks," *BioData mining*, vol. 4, no. 1, pp. 1-27, 2011.

[5] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines*: Springer, 2003, pp. 144-158.

[6] J. Wen, X. Fang, Y. Xu, C. Tian, and L. Fei, "Low-rank representation with adaptive graph regularization," *Neural Networks*, vol. 108, pp. 83-96, 2018.

[7] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616-1637, 2018.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4-24, 2020.

[9] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75-174, 2010.

[10] S. Sarkar and A. Dong, "Community detection in graphs using singular value decomposition," *Physical Review E*, vol. 83, no. 4, p. 046114, 2011.

[11] A. Zakrzewska and D. A. Bader, "A dynamic algorithm for local community detection in graphs," in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2015: IEEE, pp. 559-564.

[12] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 3, pp. 355-369, 2007.

[13] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *2016 IEEE 32nd international conference on data engineering (ICDE)*, 2016: IEEE, pp. 1218-1229.

[14] C. C. Noble and D. J. Cook, "Graph-based anomaly detection," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 631-636.

[15] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, vol. 17, no. 3, pp. 235-255, 2002.

[16] Y. Kou, C.-T. Lu, S. Sirwongwattana, and Y.-P. Huang, "Survey of fraud detection techniques," in *IEEE International Conference on Networking, Sensing and Control, 2004*, 2004, vol. 2: IEEE, pp. 749-754.

[17] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1-19, 2019.

[18] C. Chen, Z. Xu, W. Hu, Z. Zheng, and J. Zhang, "FedGL: Federated graph learning framework with global self-supervision," Information Sciences, vol. 657, pp. 119976, 2024.

[19] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1-210, 2021.

[20] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[21] B. Wang, A. Li, H. Li, and Y. Chen, "GraphFL: A federated learning framework for semi-supervised node classification on graphs," *arXiv preprint arXiv:2012.04187*, 2020.

[22] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, vol. 13, no. 1, p. 3091, 2022.

[23] S. Ru, B. Zhang, Y. Jie, C. Zhang, L. Wei, and C. Gu, "Graph neural networks for privacy-preserving recommendation with secure hardware," in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021: IEEE, pp. 395-400.

[24] R. Liu, P. Xing, Z. Deng, A. Li, C. Guan, and H. Yu, "Federated Graph Neural Networks: Overview, Techniques, and Challenges," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[25] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1-24, 2022.

[26] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 497-504.

[27] S. Gibb, H. M. La, and S. Louis, "A genetic algorithm for convolutional network structure optimization for concrete crack detection," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018: IEEE, pp. 1-8.

[28] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840-3854, 2020.

[29] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," *arXiv preprint arXiv:1810.09502*, 2018.

[30] Z. Falahiazar, A. R. Bagheri, and M. Reshadi, "Determining parameters of DBSCAN Algorithm in Dynamic Environments Automatically using Dynamic Multi-objective Genetic Algorithm," *Journal of AI and Data Mining*, vol. 10, no. 3, pp. 321-332, 2022. [Online]. Available: www.jad.shahroodut.ac.ir. [Accessed July, 2022].

[31] Y. Zhou, Q. Ye, and J. Lv, "Communication-efficient federated learning with compensated overlap-fedavg," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 192-205, 2021.

[32] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429-450, 2020.

[33] C. Cabanes *et al.*, "The CORA dataset: validation and diagnostics of ocean temperature and salinity in situ measurements," *Ocean Science Discussions*, vol. 9, no. 2, pp. 1273-1312, 2012.

[34] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93-93, 2008.

خودکارسازی مشتریان یادگیری مشارکتی در حوزه دسته‌بندی گراف با کمک الگوریتم ژنتیک

محمد رضایی، محسن رضوانی* و مرتضی زاهدی

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شاهرود، شاهرود، ایران.

ارسال ۲۰۲۳/۱۱/۰۵؛ بازنگری ۲۰۲۳/۱۱/۳۰؛ پذیرش ۲۰۲۴/۰۲/۰۲

چکیده:

با افزایش روز افزون ارتباطات و شبکه‌های اجتماعی می‌توان با بکارگیری یادگیری گراف اطلاعات ارزشمندی از داده‌ها استخراج نمود. در گذشته از روش‌های یادگیری متمرکز برای آموزش استفاده می‌شد که به دلیل وجود مشکلاتی همچون نقض حریم خصوصی داده‌گان و هزینه بر بودن نگهداری و آموزش داده‌گان در یک محیط امکان‌پذیر نمی‌باشد. یکی از رویکردهایی که برای مقابله با مشکلات یادگیری متمرکز استفاده می‌شود، بکارگیری یادگیری غیر متمرکز می‌باشد. یادگیری مشارکتی را می‌توان یکی از معروف‌ترین قالب‌های آموزش یادگیری غیر متمرکز دانست که ضمن حفظ محرمانگی داده‌ها، با استفاده از منابع مشتریان در آموزش، به کاهش هزینه‌ها نیز می‌پردازد. در سال‌های اخیر، یادگیری مشارکتی در حوزه دسته‌بندی‌گراف، توجه‌های زیادی را به خود جلب کرده‌است و در برخی از این تحقیقات نشان داده شده است که با بکارگیری فرایادگیری مدل آگنوستیک می‌توان به بهبود عملکرد یادگیری مشارکتی پرداخت. از مهم‌ترین تاثیرات، فرایادگیری مدل آگنوستیک، می‌توان به کاهش تاثیر داده‌ها با توزیع غیریکنواخت‌خطی در عملکرد مدل اشاره کرد. در محیط واقعی، تنظیم پارامترها و ساختار مشتریان یکی از چالش‌های بکارگیری یادگیری فدرال است که ایزوله بودن و تنوع داده‌ها در مشارکتی را می‌توان از مهمترین موانع برای تنظیم پارامترها و ساختار مشتریان دانست. یکی از روش‌هایی که در تحقیقات گذشته برای مقابله با این چالش مطرح شده است، بکارگیری الگوریتم ژنتیک جهت تنظیم خودکار ساختار و پارامترها می‌باشد. روش پیشنهادی با الهام و ترکیب این ساختار در مشتریان یادگیری مشارکتی که از فرایادگیری مدل آگنوستیک استفاده می‌کند، به پیشنهاد الگوریتمی که قابلیت تنظیم خودکار مشتریان را دارد، می‌پردازد. الگوریتم ژنتیک قابلیت تنظیم بخش‌های مختلف مشتری همچون ساختار دسته‌بندی گراف، نرخ یادگیری، نوع تابع بهینه‌ساز و به صورت کلی تنظیم ساختار فرایادگیری و دسته‌بندی گراف را به یادگیری مشارکتی می‌بخشد. نتایج بدست آمده از این تحقیق نشان می‌دهد، تنظیم پارامترهای مشتریان سبب بهبود دقت یادگیری مشارکتی در سطح مشتری و کارگزار می‌شود.

کلمات کلیدی: یادگیری مشارکتی، فرایادگیری مدل آگنوستیک، یادگیری مشارکتی در حوزه گراف، تنظیم خودکار مشتریان.