MG²FL: Multi-Granularity Grouping-Based Federated Learning in Green Edge Computing Systems

Ziming Dai*‡, Yunfeng Zhao*, Chao Qiu*‡, Xiaofei Wang*, F. Richard Yu†,
*College of Intelligence and Computing, Tianjin University, Tianjin, China
†School of Information Technology, Carleton University, Ottawa, ON, Canada
‡Guangdong Laboratory of Artificial Intelligence and Digital Economy, Shenzhen, China
Email: {dzm1179838, yfzhao97, chao.qiu, xiaofeiwang}@tju.edu.cn, richard.yu@carleton.ca

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437153

***Abstract*—Federated Learning (FL) has become a common method for edge devices. Due to the limited energy capacity of edge devices, and the vulnerability of FL to malicious attacks from edge devices, vanilla FL still faces several challenges in edge computing, including energy consumption, model heterogeneity, and malicious behavior. To address these challenges, we propose a multi-granularity grouping-based federated learning (MG²FL), which groups and aggregates edge devices with low communication energy consumption and latency to reduce communication costs. Additionally, we introduce a multi-granularity guidance mechanism and a credit model to enhance model accuracy while ensuring security. Experimental results show that compared to the traditional FL algorithms, MG²FL achieves a 5.6% increase in accuracy, with the highest accuracy improvement reaching 11.1% in the presence of malicious edge devices.**

***Index Terms*—Edge devices, federated learning, balanced graph partitioning, multi-granularity guidance, credit model.**

I. Introduction

With the emergence of edge devices such as unmanned aerial vehicles (UAVs) and mobile robots, these edge devices have found extensive applications in various fields such as scene understanding, emergency search, and urban target tracking [1]–[4]. To achieve autonomous operations in these applications, edge devices in the scene need to make real-time decisions, based on intelligent analysis of collected data. Due to the high requirements of bandwidth and latency for raw data transmission, traditional centralized machine learning approaches are not suitable for real-time applications on edge devices [5]. In contrast, federated learning (FL), as a distributed machine learning method, is more suitable for edge devices since it transmits parameters, rather than raw data [6].

However, vanilla FL still faces several challenges, especially in the field of edge computing. ***(i) Energy consumption.*** Communication between certain edge devices leads to high energy consumption. The participation of a large number of edge devices in FL increases the frequency of communication, leading to higher energy consumption, thereby causing environmental degradation. ***(ii) Model heterogeneity.*** The datasets used by edge devices for different tasks may have different granularities, and the models on different edge devices may also vary. This heterogeneity increases the difficulty of model aggregation. ***(iii) Malicious behavior.*** Due to the openness of edges, there is a possibility of malicious behavior where edge devices upload incorrect parameters, which undermines the accuracy of the model aggregation.

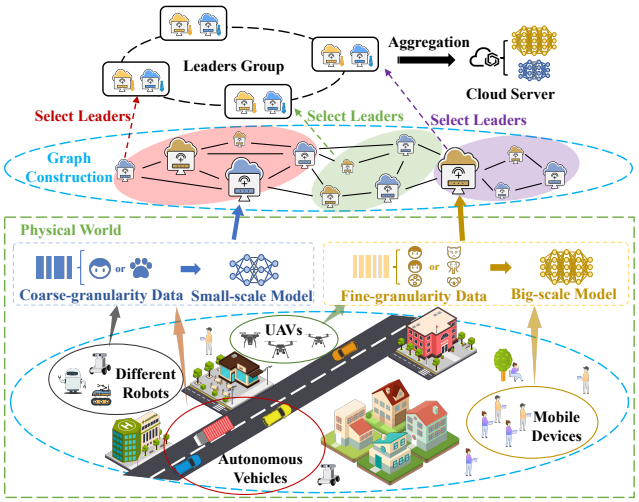


Fig. 1: MG²FL in green edge computing systems.

Some papers have focused on reducing FL energy consumption, such as Yang *et al.* in [7] proposed an iterative algorithm to minimize the total energy consumption of local computation and wireless transmission. Quoc-Viet Pham *et al.* in [8] introduced the E2FL algorithm for energy reduction in FL among drones. Others have considered only the heterogeneity of models or datasets, such as Cai *et al.* in [9] proposed a multi-granularity guidance FL to enhance performance. The above works didn't solve the previous three challenges comprehensively.

In this paper, we propose a multi-granularity grouping-based federated learning approach (MG²FL). The main contributions of this paper are summarized as follows:

- Considering reducing communication energy consumption and latency (referred to as communication overhead in the following text), the MG²FL utilizes balanced graph partitioning to group edge devices. Further, we design a guiding method among models with different granularities to enhance model performance.

- The MG²FL incorporates a credit model, which considers dynamically changing credit scores and selects the edge devices with the highest credit score as leaders for each group. The global model is then aggregated by the leaders. This approach significantly improves security.
- Extensive experiments demonstrate that MG²FL improves models' accuracy by 2.7% to 5.6%. Our approach strikes an effective balance between model performance and model overhead, resulting in an improvement of 3.9% to 11.1% on the final trade-off objective.

II. System Architecture and Model

A. System Architecture

Fig. 1 shows the framework of MG²FL, which contains multiple edge devices $\mathcal{E} = \{E_1, E_2, ..., E_n\}$. The scale of models and granularity of datasets owned by these edge devices could be heterogeneous. In this paper, we consider two different models and two different granularity levels of datasets. The models are denoted as $\mathcal{M} = \{M_1, M_2, ..., M_n\}$, which can be divided into big-scale model $\mathcal{M}_b$ and small-scale model $\mathcal{M}_s$. Similarly, the parameters of the model are represented as $\mathcal{W} = \{w_1, w_2, ..., w_n\}$.

Generally, larger-scale models often have higher learning capabilities, while smaller-scale models exhibit the opposite trend [9]. Therefore, we divide all edge devices into two types: small-scale edge devices with coarse-granularity data $\mathcal{E}_c$ and large-scale edge devices with fine-granularity data $\mathcal{E}_f$.

B. Communication Latency Model

During each iteration of FL, all edge devices upload their local training model parameters to the central server, while the FL server broadcasts the global model to each edge device. Communication latency is inevitable during the process of uploading and downloading data. Communication latency is generally divided into propagation latency and transmission latency, and the transmission latency is related to the size of the data model and the transmission power, which will be considered when calculating the transmission energy consumption in the next section. Therefore, we only consider propagation latency in communication latency. The propagation latency from edge device E_i to edge device E_j is calculated as $t_{ij}^{latency} = d_{ij}/v$, where d_{ij} is the distance between edge device E_i and edge device E_j, and v represents the propagation speed of the signal [10].

C. Transmission Energy Consumption

In FL, data transmission leads to energy consumption. From the Shannon formula [11], we express the data transmission rate between edge device E_i and E_j, i.e. r_{ij}, as

$$r_{ij} = B_{ij} \log_2(1 + \frac{g_{ij} p_{ij}}{N_0 B_{ij}}), \tag{1}$$

where B_{ij} represents the bandwidth between two edge devices, N_0 is the power spectral density of Gaussian noise, g_{ij} is the channel gain between edge device E_i and E_j, and p_{ij} denotes the transmission power. Communicating operations will happen in model aggregation and model guidance. If the transmission data size is d, the transmission time is $T_{ij} = d/r_{ij}$. Then the transmission energy consumption between edge device E_i and edge device E_j is $E_{ij}^{trans} = p_{ij} T_{ij}$.

D. Guidance Ability Model

As mentioned in Section II-A, there are two types of edge devices, and we consider the higher granularity models as guides, which guide the models trained with lower granularity data to improve their performance.

We quantify the guiding ability of edge device E_i on edge device E_j and represent it using the symbol π_{ij}:

$$\begin{aligned} \pi_{ij} &= \varphi(w_i, x_j, y_j) - A_j, \\ \varphi(w_i, x_j, y_j) &= \frac{\sum_{k=1}^{|x_j|} \mathbb{I}_{\{H \cdot p(w_i, x_{j,k}) = y_{j,k}\}}}{|x_j|}, \end{aligned} \tag{2}$$

where $p(w_i, x_{j,k})$ denotes the output of model M_i on the input data $x_{j,k}$. $y_{j,k}$ represents the k-th label of edge device E_j and H is the knowledge matrix that stores the relationship between two types of data. It makes transforming one type of data to another through using matrix H possible. $\varphi(w_i, x_j, y_j)$ is the conversion accuracy of edge device E_i on the data x_j. The function $\mathbb{I}(\cdot)$ is the indicator function and A_j indicates the accuracy of edge device E_j on the local dataset.

E. Credit Model

In order to reduce the risk of malicious edge devices providing fake quality assessment results, we score each edge device. This score determines the weight of the edge device in the global aggregation, the higher the score, the more important its model parameters are. And the edge device with the highest score will be selected as the leader to host model aggregation. Specifically, we name this score as the credit score of the edge device, which is related to the data size, computing power of the edge device and model accuracy. The credit score of edge device E_i is described as:

$$C_i = \log \frac{D_i}{\sum_{i=1}^{n} D_i} + f_i + c_i^I, \tag{3}$$

where $c_i^I = \sum_{k=1}^{I} 1/\{1 + e^{-\log(c_i^{k-1} + a_i)}\}, \quad c_i^0 = 0$. D_i, f_i, and I represent the data size, computing power, and current iteration number of edge device E_i, respectively. The accuracy of testing the model of i by the leader is denoted as a_i.

It is noteworthy that the latest credit score is related to its learning performance and accumulative credit score. To achieve normalization in the credit score calculation, a sigmoid function is employed. If an edge device performs well in previous training iterations, it will not be discarded due to poor performance in a single iteration. When the edge devices in every group send their model parameters to the group leader, it will validate the local updates with the test dataset. Afterward, the edge device's credit score will be obtained by the validation accuracy and its historical credit score.

F. Problem Formulation

In this paper, we investigate multi-granularity data FL with energy and latency requirements under security constraints. We aim to reduce the communication overhead of FL through grouping while improving the model accuracy using multi-granularity guidance, all while adhering to security constraints.

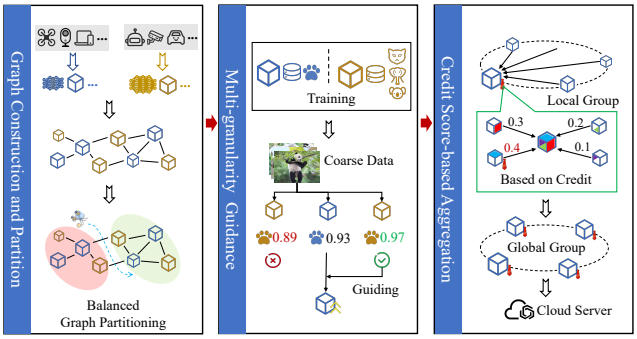


Fig. 2: The framework of multi-granularity grouping-based FL.

Our ultimate goal is to achieve an ideal model that strikes a balance between performance and overhead. We illustrate the problem by taking the k-th group s_k as an example.

$$\max \frac{1}{|s_k|} \sum_{i \in s_k} [A_i - \Pi(i,j)] \quad s.t.\ j \in N(i), \tag{4}$$

where $\Pi(i,j)$ means the function that calculates the communication overhead between edge device E_i and E_j, and $N(i)$ refers to all the connected edge devices to the edge device E_i.

To further simplify the objective function, we divide it into two parts. First, edge devices with low communication latency and energy consumption and high mutual guidance ability are grouped into a group through a partitioning algorithm. Meanwhile, in order to ensure the training duration of each group, the partitioning should be as even as possible. We employ the partitioning method to achieve the following objectives:

$$\min \sum_{s_k \in \mathcal{S}} \sum_{i,j \in s_k} e_{ij} \quad s.t. \bigcup_{s_k \in \mathcal{S}} s_k = \mathcal{E}, \bigcap_{s_k \in \mathcal{S}} s_k = \emptyset, \tag{5}$$

$$\max_{s_k \in \mathcal{S}} |V_{s_k}| \leq (1+\varepsilon) \frac{\sum_{s_k \in \mathcal{S}} |V_{s_k}|}{|\mathcal{S}|}, \tag{6}$$

where e_{ij} denotes the weight of the edge between E_i and E_j, and V_{s_k} represents the sum of the weight of all the edge devices within the group s_k. The $\mathcal{S}$ in the equation means the groups obtained after partitioning through the algorithm. The ε is a hyperparameter to tune the imbalance of partition.

Furthermore, the model performance is enhanced within each group using multi-granularity guidance. The training process within the s group can be represented as follows:

$$\begin{aligned} &\min \sum_{i \in s} [\frac{1}{|x_i|} \sum_{k=1}^{|x_i|} F_i(M_i, x_{i,k}, y_{i,k}) + \beta \zeta(M_i, M_j)], \\ &\zeta(M_i, M_j) = \frac{\sum_{r=1}^{|x_i|} ||\sigma(M_i, x_{i,r}) - \sigma(M_j, x_{i,r})||^2}{|x_i|}, \end{aligned} \tag{7}$$

where $\sigma(\cdot)$ is the output of Deep Neural Networks (DNNs) before the last dense layer, $\zeta(M_i, M_j)$ calculates the difference between the two models. $F_i(\cdot)$ is the loss function of E_i.

III. MG²FL Procedure

To reduce communication overhead in FL on heterogeneous edge devices, we have developed a new collaborative learning framework: MG²FL. More detailed information about the MG²FL operation process is depicted in Fig. 2.

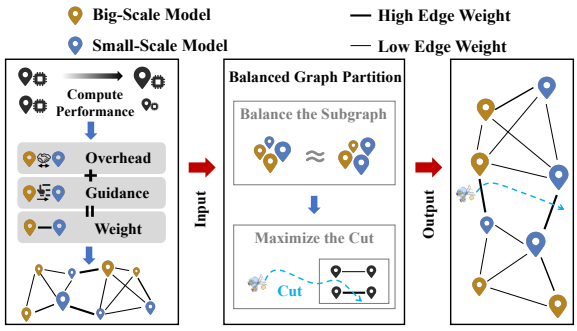


Fig. 3: The graph construction and partition.

A. Test Training

Before grouping the edge devices, we need to have knowledge of the performance metrics of the edge devices as well as their guiding abilities between devices. At the beginning of the system process, each edge device will train its model using a local test dataset. Due to the limited size of the test data, the model training process can be completed quickly. The training time for edge device E_i will be stored on the device and denoted by t_i.

After the completion of model training, edge devices that can communicate with each other will engage in model guidance. To improve the performance of the model, we only allow fine-granularity models to guide coarse-granularity models. For ease of storage, the guidance ability parameters will be stored in the guided edge device. Specifically, edge device E_j will store the guidance ability parameters π_{ij} from adjacent edge device E_i that provided guidance to E_j.

B. Graph Construction

As shown in Fig. 3, we model the edge devices network as a weighted undirected graph G with nodes consisting of $\{\mathcal{E}_c, \mathcal{E}_f\}$ and edges $\{e_{12}, e_{13}, ..., e_{(n-1)n}\}$, where edges denote the communication overhead between edge devices.

Node weight: In order to minimize the communication overhead of the system and ensure high guidance capability, we hope to achieve load balancing among different groups. Hence, we set the weights of nodes as the training time of each edge device. However, using only the training time of the first test is not accurate. Therefore, in the setting of node weights, we also consider the size of the edge device dataset and its hardware performance. So we define the weight of edge device E_i in the graph as $W_i = (\gamma t_i + \frac{1}{1+e^{-D_i/f_i}})/2$, where γ is used to tune the influence of training time.

Edge weight: After grouping, we hope that the communication cost between edge devices within a group is low, and the guidance ability between them is high. So, the weight of edges e_{ij} in the graph is defined as the weighted sum of latency, energy consumption, and guidance ability between the edge devices within a group.

$$e_{ij} = \nu \frac{1}{\pi_{ij}} + \varsigma t_{ij}^{latency} + \tau E_{ij}^{trans}, \ i > j \tag{8}$$

where ν, ς and τ are hyperparameters that adjust the relative importance of three distinct attributes.

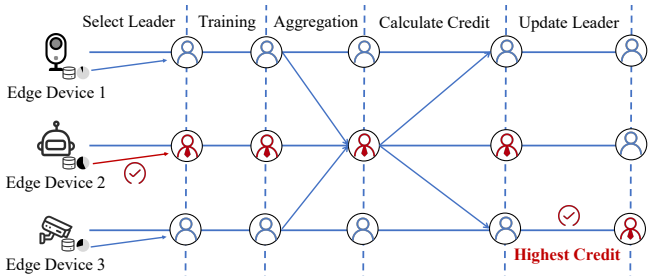


Fig. 4: The process of leader updating in MG²FL.

C. Graph Partitioning

After defining the graph G, we adopt the balanced graph partitioning algorithm KaHyPar [12]. It is a multi-level graph partitioning framework that utilizes graph coarsening, initial partitioning, and local search algorithms. The goal of this partitioning algorithm is to achieve a balanced partitioning among groups while cutting the edges with the highest weights. By introducing the definition of edge weights, this algorithm ensures that edges with weak guiding abilities and high communication overhead are reduced, thus ensuring that the abilities and overhead among edge devices within each group meet the optimization objectives.

We will calculate the credit score of each edge device according to Eq. (3), and temporarily select the edge device with the highest credit score as the leader, responsible for global graph partitioning.

D. Multi-Granularity Guidance in FL

The initial model parameters are stored on each edge device.

- Local model training: Based on the initialized model parameters, edge devices access the latest model and execute local training. Specifically, the edge device in every training group will train the local model based on its raw data so that the expectation of a mini-batch from every edge device's local dataset is minimized.
- Local model guidance: After completing the local model training, each edge device E_i will select the edge device with the highest π_{ij} from its local storage for model guidance, thus improving the model performance. The edge device E_i performs parameters updates as:

$$\begin{aligned} & w_i = w_i - \eta \bigtriangledown \zeta(M_i, M_j), \\ s.t. \quad & j = \arg\max \ \pi_{ij}, \quad j \in N(i), \end{aligned} \tag{9}$$

  where η means the learning rate in the update process. This process only occurs in the later stage of the system, aiming of ensuring that the models used for guidance are mature enough to guide models of guided edge devices.

E. Leader Selection Based on Credit Score

As shown in Fig. 4, during the local model aggregation process, the leader of group s_k performs weight model aggregation locally to obtain the local model parameters w^{s_k}, where the weight of each edge device in the model aggregation depends on its credit score, like Eq. (10). The initial credit score of each edge device is only related to the size of its local dataset, but the credit score of each edge device will be recalculated in each iteration, as shown in Eq. (3).

$$w^{s_k} = \sum_{j \in s_k} C_j w_j, \quad s_k \in \mathcal{S}. \tag{10}$$

Assuming local aggregation is synchronous, the aggregation will be triggered when enough edge devices upload their local models to the leader. Once local model aggregation is completed, all local aggregation parameters will be uploaded to the global model aggregation group to obtain global updates. Due to the fact that our system has two models, we will select two leaders for aggregation, with each leader being the edge device with the highest credit score among those with the same model in the group.

Before global aggregation, each group has two leaders, each responsible for storing a different type of model. In global aggregation, we divide all group leaders into two groups according to their model types, and elect the edge device with the highest credit score in each group as the leader of the global aggregation group. These two leaders perform global aggregation locally and generate a new global model.

At the end of each iteration, a new local aggregation leader is selected. This ensures that each edge device has the potential to become the leader, provided that the edge device has the highest credit score. The FL mechanism is implemented through local model aggregation by the leader. If the leader's behavior is malicious, the performance of the global model will be greatly affected.

F. Global Model Aggregation and Group Leader Updating

In MG²FL, the edge device with the highest credit score is selected as the leader, and all leaders form a new global

Algorithm 1: MG²FL Training process

Input: All edge devices' initial local model's parameters w_i.
Output: Global model's parameters w^G.

```
for each group s_k do
    for e = 1, 2, ..., E do
        for each edge device E_i, i ∈ s_k do
            E_i solves the local problem and derives w_i.
            Transmit w_i and A_i to the leader device.
        end
        After receiving w_i and A_i, get C_i by Eq. (3).
        The leader chooses the edge device with the
         highest C_i as the new leader.
        if (e ≥ β_f) and (e%θ == 0) then
            Calculate w_i by Eq. (9).
        end
        else
            Calculate w^{s_k} by Eq. (10).
        end
        The leader broadcasts w^{s_k} to edge devices.
    end
end
Each group's leader aggregates to obtain w^G.
```

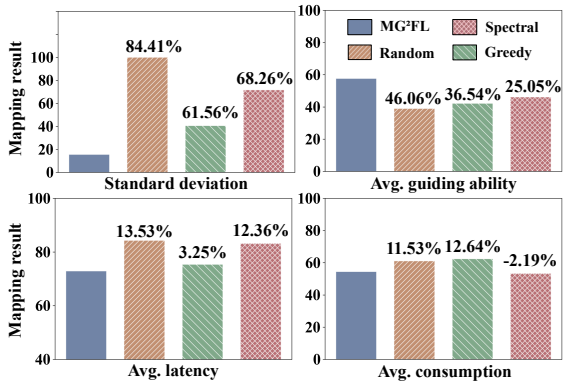
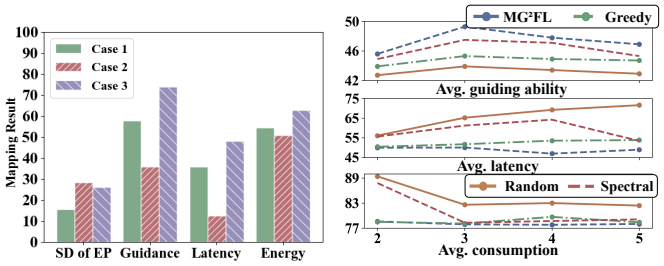


Fig. 5: Evaluation of different grouping method.



(a) The influence of different attention on the grouping results.

(b) The influence of different numbers of groups on the model.

Fig. 6: The influence of hyperparameters on the model results.

aggregation group. Since the credit score of edge devices considers both physical factors (i.e., data size) and training performance, the leader update process abandons malicious edge devices, further increasing the cost of attacks and enhancing security and stability. The training process of MG²FL is shown as Algorithm 1, where θ represents the frequency of guided operations during the guidance period, E is the epoch number and β_f denotes the starting epoch of guided operations.

IV. Performance Evaluation

In this section, we evaluate the MG²FL framework's performance, focusing on: performance and communication overhead under different grouping schemes, the improvement brought by guidance and the impact of malicious edge devices.

A. Dataset and Experiment Settings

In our experimental study, we employed the CIFAR100 dataset [13], which is characterized by two types of labels for each sample. The dataset classifies common objects into 100 fine-granularity classes and 20 coarse-granularity classes, with each coarse-granularity class including five fine-granularity classes. We utilized the Wide-Resnet DNN model with multiple convolutional layers for image classification. The batch size was 64, and the local iterative number was set at 5. In order to provide a more intuitive representation of the evaluation metrics in our experiments, we map the evaluation metrics to a range of 0 to 100 as mapping result.

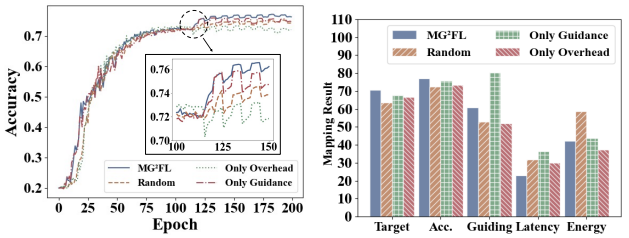
B. Experiment Results

1) Grouping method analysis: We evaluate the impact of different grouping methods, including random grouping, greedy grouping, spectral clustering [14], and MG²FL, on various system metrics. The grouping results are depicted in Fig. 5, which presents four evaluation results: standard deviation of computational power between groups, average guiding ability, average latency, and average energy consumption. From the graph, it can be observed that MG²FL ensures similar computational capabilities among different groups while achieving a minimum improvement of 25% in guidance capability. Additionally, compared to methods with similar latency, MG²FL reduces energy consumption by 12.64%. This can be attributed to the graph partitioning technique, which divides the system into groups by cutting high-weighted edges. By defining edge weights, this approach takes into account the guidance capability, energy consumption and latency among edge devices simultaneously.

2) Hyperparameters analysis: To meet these different requirements for the model, we introduce hyperparameters in Eq. (8) to dynamically adjust the attention of the model. In Fig. 6 (a), we present Case 1, which balances performance and overhead, with hyperparameters set as $\nu = 0.3$, $\varsigma = 0.3$, $\tau = 0.3$; in Case 2, which prioritizes overhead, hyperparameters are set as 0.2, 0.4, and 0.4; and in Case 3, which emphasizes performance, hyperparameters are set as 0.6, 0.2, and 0.2. As seen from the figure, we evaluate the standard deviation of edge device performance (SD of EP), the average guidance, latency and energy of the system for each group. We observed that changes in weights affect the mapping result of various metrics. Attributes that acquire higher weights tend to yield better mapping result for those attributes.

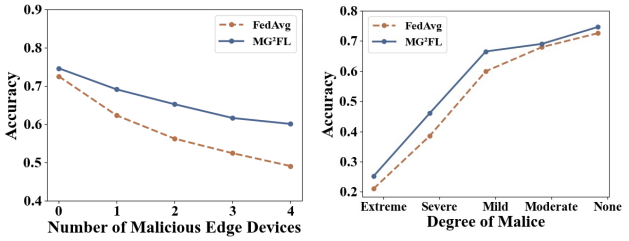
We analyzed the impact of different numbers of groups on the system, as shown in Fig. 6 (b). We found that regardless of the number of groups, MG²FL consistently outperformed the other methods. Specifically, the guiding capability initially increased and then decreased with an increase in the number of groups. This is because when the number of groups becomes too large, the number of models of different granularities within each group becomes too low, leading to a decrease in guiding capability. However, energy consumption and latency decreased with an increase in the number of groups. This is because the larger the number of groups, the finer the partitions, and thus more costly communication is forsake.

3) Performance analysis: Considering the significant impact of multi-granularity guidance on model performance under different grouping approaches, we analyze the performance curves and evaluate other metrics. Fig. 7 (a) depicts the impact of different grouping approaches on the guiding performance of the MG²FL model. Firstly, it can be observed that we take full advantage of heterogeneity, improving the performance of the model by 6% using guidance at an epoch of 110. Furthermore, it can be seen that the final performance of MG²FL is 2.7% to 5.6% higher than the baseline. This is because our grouping method strives to place edge devices with strong guidance capability for E_i in the same group as E_i while ensuring weight balance between groups, avoiding the issue of decreased guidance effectiveness due to a shortage



(a) Performance comparison. (b) Evaluation comparison.

Fig. 7: Guidance from fine to coarse-granularity edge devices.



(a) Number of malice. (b) The degree of malice.

Fig. 8: Impact of malicious edge devices on MG²FL.

of edge devices in a group.

We further analyze the results, as shown in Fig. 7 (b), where target is the evaluation metric to trade off accuracy, communication overhead, and guidance. The greedy grouping approach often exhibits an advantage in a specific metric. Taking the grouping method that only considers overhead as an example, although it has the lowest average energy consumption, it fails to account for the guiding ability among edge devices. Consequently, its model accuracy after guidance is lower, resulting in its inferior performance compared to the MG²FL framework in evaluation metrics that consider both model performance and overhead.

4) Security analysis: We simulate attacks from malicious edge devices from two perspectives: the number of malicious edge devices and the degree of their malicious behavior. Fig. 8 (a) and Fig. 8 (b), respectively, show the impact of the two malicious approaches on the performance of the proposed MG²FL and the traditional FedAvg [15]. We observe that in the presence of malicious edge devices, MG²FL consistently outperforms FedAvg. Moreover, the performance gap between MG²FL and FedAvg widens as the number of malicious nodes or the severity of malicious behavior increases, with the largest difference between the two methods reaching 11% and 6.6%.

After analysis, we attribute this result to the following reason: When the number of malicious edge devices or the severity of their behavior is low, their contribution to the overall aggregation in FedAvg is minimal due to their low weight. However, as the malicious behavior or the number of edge devices increases, traditional FedAvg becomes more susceptible to the influence of malicious nodes, while in MG²FL, the credit score ensures that weights of malicious edge devices decrease, thereby preserving the model performance.

V. Conclusion

In this paper, we address the issue of high overhead in FL among multi-granularity edge devices, enabling these edge devices to efficiently perform heterogeneous FL tasks with limited power resources and enhanced security. We propose a novel method called MG²FL, which combines balanced graph partitioning and multi-granularity guidance. Experimental results demonstrate that our approach outperforms other baselines in terms of performance and communication overhead. Additionally, our method uses a credit model to mitigate the impact of malicious edge devices on the FL results.

VI. Acknowledgement

This work was supported in part by China NSFC (Youth) through grant No. 62002260, in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grant GML-KF-22-03, in part by China NSFC through grant No. 62072332, in part by the Tianjin Xinchuang Haihe Lab under Grant No.22HHXCJC00002.

References

[1] A. Jokić, M. Petrović, and Z. Miljković, "Real-time mobile robot perception based on deep learning detection model," in *New Technologies, Development and Application V*, 2022, pp. 670–677.

[2] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Commun Mag*, vol. 54, no. 5, pp. 36–42, 2016.

[3] R. Song, T. Long, Z. Wang, Y. Cao, and G. Xu, "Multi-UAV cooperative target tracking method using sparse a search and standoff tracking algorithms," in *IEEE CGNCC*, 2018, pp. 1–6.

[4] J. Xie, Z. Chang, X. Guo, and T. Hämäläinen, "Energy efficient resource allocation for wireless powered uav wireless communication system with short packet," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 101–113, 2023.

[5] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, 2020.

[6] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial internet of things," *IEEE Trans Industr. Inform.*, vol. 17, no. 8, pp. 5605–5614, 2020.

[7] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 3, pp. 1935–1949, 2020.

[8] Q.-V. Pham, M. Le, T. Huynh-The, Z. Han, and W.-J. Hwang, "Energy-efficient federated learning over UAV-enabled wireless powered communications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4977–4990, 2022.

[9] S. Cai, Y. Zhao, Z. Liu, C. Qiu, X. Wang, and Q. Hu, "Multi-granularity weighted federated learning in heterogeneous mobile edge computing systems," in *IEEE ICDCS*, 2022, pp. 436–446.

[10] D. Zhao, T. Yang, Y. Jin, and Y. Xu, "A service migration strategy based on multiple attribute decision in mobile edge computing," in *IEEE ICCT*, 2017, pp. 986–990.

[11] X. Zhou, J. Zhao, H. Han, and C. Guet, "Joint optimization of energy consumption and completion time in federated learning," in *IEEE ICDCS*, 2022.

[12] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders, "High-quality hypergraph partitioning," *ACM J. Exp. Algorithmics*, 2022.

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

[14] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, 2007.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.