Cluster Computing (2025) 28:400
https://doi.org/10.1007/s10586-025-05106-5

Optimizing Client Selection in Federated Learning Base on Genetic Algorithm

Jiagao Wu[1,2] · Hongyan Ji[1,2] · Jing Yi[1,2] · Linfeng Liu[1,2]

Received: 13 June 2024 / Revised: 30 December 2024 / Accepted: 3 January 2025 / Published online: 16 June 2025

Abstract
Federated Learning (FL) significantly advances the field of distributed machine learning by allowing devices to collaboratively train models without the need to exchange sensitive local data, thus maintaining privacy and security. However, FL faces serious challenges in heterogeneous network environments where devices vary greatly in system resources (system heterogeneity) and data distributions (data heterogeneity), which impact the efficiency and effectiveness of the model training. To address these issues, we present FedCSGA, a novel FL client selection algorithm based on the genetic algorithm with adaptive genetic operators to maximize the number of selected clients within a training time deadline per round. Furthermore, we introduce the model accuracy into the fitness function of FedCSGA and enable the algorithm to handle the system and data heterogeneity simultaneously. Our comprehensive experiments demonstrate that the proposed FedCSGA algorithm can significantly outperform several state-of-the-art baselines and increase the number of clients selected per round by 45.6%, 54.3%, the model accuracy by 2.4%, 7.7% on average in both IID and non-IID settings, respectively.

Keywords Federated learning · distributed machine learning · heterogeneity · client selection · genetic algorithm

1 Introduction

In traditional cloud-centric machine learning, data from terminal devices (e.g., personal health data, financial information, communication records, etc.) are collected and sent to a cloud-based server or data center for processing and training [1, 2]. However, in the era of the Internet of Things (IoT), different IoT devices (e.g., wireless sensors, smartphones, vehicles, etc.) generate large amounts of data every moment, and uploading all of these data to the cloud server will consume significant network bandwidth. Additionally, the cloud-centered learning approach has high transmission latency and cannot facilitate timely data exchange, which obstacles the deployments of many network applications.

Nowadays, the primary source of data comes from edge-based devices rather than cloud-based devices. Hence, Mobile Edge Computing (MEC) is naturally proposed as a solution, where the computation and storage capabilities of mobile devices and MEC servers are leveraged to push computation tasks from the center to the edge of the network that is closer to users and data sources [3, 4]. In particular, by offloading computation-intensive and latency-sensitive tasks (i.e., autonomous driving, object detection, etc.) to the resource-rich MEC servers, MEC can significantly improve latency performance and reduce the energy consumption of mobile devices, and thus support large-scale distributed tasks and applications.

Federated Learning (FL) is a novel distributed machine learning framework proposed in recent years [5]. During

✉ Jiagao Wu
jgwu@njupt.edu.cn

Hongyan Ji
1221045820@njupt.edu.cn

Jing Yi
1022040912@njupt.edu.cn

Linfeng Liu
liulf@njupt.edu.cn

1 School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

2 Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

the training process of FL, only the local models trained by all clients (e.g., mobile devices) on their private data are uploaded to a server (e.g., MEC server) for aggregation without involving the data itself and greatly improving user data privacy. This computational pattern of FL is naturally consistent with that of the MEC framework. Besides, training the model on mobile devices and keeping the data local also saves communication costs and protects privacy. Therefore, FL has attracted a lot of interest from researchers [6–8] and has been widely applied in practice [9–11] since it was proposed.

Generally speaking, FL iteratively requires the server to distribute global model parameters to a set of clients, which train the model locally using their own data, and then upload the new model parameters to the server, while the server aggregates the updated model parameters from clients to further improve the model. The most typical FL aggregation algorithm is the Federated Average (FedAvg) algorithm [5], which requires clients to perform multiple rounds of local training before uploading their model parameters to the server, and then take the average of these models to generate a better global model.

However, unlike training in the data center where data is sufficient and resources are unrestricted, the storage, computing, and communication capabilities of each client participating in FL may vary depending on their hardware configurations, network connections, and power availabilities, which lead to system heterogeneity [12]. Additionally, as the local data on each client typically depends on user preference and usage, the data distributions on clients often exhibit non-Independence and Identically Distributed (non-IID) characteristics, which lead to data heterogeneity [13]. These factors significantly impact the training time and model accuracy of FL [14]. Thus, selecting suitable clients to participate in FL can effectively improve the training efficiency of FL and is one of the hottest key research points for FL optimization.

Existing client selection methods in FL can be mainly categorized into two classes: optimization-based and utility-function-based. For the first class, an optimization problem usually is formulated, and then, a heuristic client selection algorithm is designed to solve the problem [15–21]. However, the complexity of these optimization problems leads to a major challenge in designing optimization algorithms, and the performance of the existing heuristic or greedy algorithms is limited. Hence, another kind of client selection method based on the utility function is proposed, where clients with higher utility have more chances to be selected per training round [22–24]. Although this approach is simple, the utility functions are usually defined by empirical formulas and lack the support of theories and models.

To this end, in this paper, we adopt the client selection approach based on the optimization problem with the objective of maximizing the number of clients under the constraint of limited training time per round. Specifically, regarding the NP-hardness of the problem and the limitations of existing heuristic algorithms, other novel computational intelligence paradigms called metaheuristic algorithms should be adopted here [25]. There are many classic metaheuristic algorithms such as genetic algorithms, particle swarm optimization algorithms, differential evolution algorithms, etc. All of them can solve complex optimization problems by combining intelligently different strategies for exploring and exploiting the search space to find efficiently near-optimal solutions. The particle swarm optimization algorithm and the differential evolution algorithm are more suitable for continuous optimization problems, while the genetic algorithm is more suitable for discrete optimization problems. Hence, we optimize the client selection in FL based on *genetic algorithm* [26]. The genetic algorithm is one of the well-known evolutionary algorithms, which is characterized by the elements of chromosome representation, fitness function, and a set of genetic operators (i.e., crossover, mutation, and selection operators). In this way, genetic algorithms can efficiently search the global optimum in a multi-dimensional space and thus have been successfully and broadly applied to solve combinatorial optimization problems. Additionally, we consider both system and data heterogeneity to improve the learning efficiency and convergence speed of FL.

The main contributions of this paper are as follows:

1) To overcome the limitations of existing heuristic and greedy algorithms, we propose a novel client selection algorithm for FL based on the genetic algorithm, called FedCSGA, which aims to maximize the number of selected clients within a specified training time deadline per round. By adaptive crossover, mutation operators, and fine-tuning parameters, FedCSGA can obtain an approximate optimal solution and increase the number of clients selected per round.
2) To solve the non-IID issues, we further introduce the term model accuracy into the fitness function of FedCSGA. In this way, FedCSGA can handle both system and data heterogeneity of clients to enhance the training performance of FL effectively and avoid leaking the privacy information of client data distribution as well.
3) The extensive experiments show that FedCSGA can significantly increase the number of clients selected per round by 45.6%, 54.3%, the model accuracy by 2.4%, 7.7% on average compared with several baseline FL algorithms in both IID and non-IID

settings, respectively. Thus, the effectiveness of the proposed optimization methods in FedCSGA is verified.

2 Related work

Google introduced the concept of FL [3], which facilitates distributed machine learning among devices while remaining data on individual devices. In FL, the training task of the model is decentralized to mobile devices at the edge of the network, so the system and data heterogeneity for the devices (i.e., clients) is a crucial factor affecting FL training. Selecting suitable clients to participate in the training and effectively reduce the training time is an important research problem in FL.

For the optimization-based client selection methods, according to the different optimization objectives, these methods can be further divided into two sub-categories: maximizing the number of participating clients [15–18] and minimizing the training time per round [19–21]. Nishio et al. [15] considered that clients with limited computational resources or poor wireless channel conditions would require longer training or uploading time in FL. To address this issue, the authors modeled the FL training process as an optimization problem and proposed the FedCS protocol. This protocol collects resource information of each client in advance and uses a greedy algorithm to select clients with better computation or communication conditions, which makes the server aggregate as many client updates as possible within the time deadline for each round and effectively improves the convergence speed of the model. However, FedCS does not consider the problem of data heterogeneity. Furthermore, Yoshida et al. [16] proposed a hybrid learning mechanism called Hybrid-FL, in which the server constructs partially the IID data using data collected from a limited portion of clients (e.g., less than 1%) with low privacy sensitivity, and trains the model on the constructed data, and aggregates it with the clients' trained models for global model updating. To improve the accuracy of the global model, the authors also designed a heuristic algorithm to select the optimal client set according to both the training time and the data distribution of clients under the time deadline constraints each round. The problem of Hybrid-FL is that clients may be reluctant to share data considering user privacy concerns. In addition, Abdulrahman et al. [17] proposed a multicriteria-based FL client selection model, where all resources of clients (e.g., CPU, memory, energy, and time) are considered restricted. Based on this, the authors utilized hierarchical sampling to filter the available client set and proposed a multi-criteria FL client selection (FedMCCS) algorithm to maximize the number of selected clients under the multiple criteria of resource constraints. Xu et al. [18] study the client selection of FL in a wireless network. They formulated a stochastic optimization problem and proposed a heuristic online algorithm to optimize the number of clients and bandwidth allocation per round under the constraint of limited client energy (e.g., a finite battery). Besides, there is another kind of client selection algorithm that minimizes the training time for each round to speed up the FL training. Shi et al. [19] proposed a strategy to decouple the optimization problem into bandwidth allocation and client selection sub-problems. For the first sub-problem, the authors designed a binary search algorithm to minimize the time needed for the round. For the second sub-problem, the authors adopted a greedy client selection strategy that consumes less time in training to achieve the best trade-off between learning efficiency and training time. In addition, Ching et al. [20] formulated a new optimization problem called TRAIN, whose aim is to minimize the training costs (e.g., incentive payments and uploading time) while ensuring the data quantity requirement. The authors proved that TRAIN is NP-hard, and proposed an approximate algorithm to find a subset of clients and schedule their uploading sequence to obtain a near-optimal upload time per round of training. Wang et al. [21] studied client selection problems under both IID and non-IID data distributions. Under the number of clients constraint, for the IID data distribution, the authors found the near-optimal solution to minimize the training latency of each round without reducing the accuracy. For the non-IID data distribution, the authors transformed it into the problem of minimizing the average cost and proposed a greedy algorithm to balance the training latency and model precision. However, most optimization problems of client selection in FL are NP-hard and cannot be solved by the greedy and heuristic algorithms efficiently. Recently, Kang et al. [27] proposed a genetic algorithm for client selection in FL. To speed up FL training, the fitness is defined as a combination of the change in the model's parameter and the communication cost of each client, that is, chromosomes of selected clients have higher fitness when they have larger updates of the model with lower communication. The communication cost is a general concept, which can be represented as the model size, client data sharing preferences, network conditions, etc. Nevertheless, in this paper, they only defined the communication cost as the reciprocal of the data size of the client and did not consider the real computation and communication delay in our system model. Besides, the volume of the change in the model's parameter is usually large, which will increase the time complexity of calculating their fitness. Therefore, although Kang et al. [27] and we all use genetic algorithms for client selection in FL, we proposed a more practical system model with a more efficient optimization method.

To avoid the difficulty of solving the optimization problem, several other client selection methods based on the utility function are proposed [22–24]. Zhang et al. [22] observed heterogeneous weight divergence between clients with varying degrees of non-IID data and clients with IID data, then proposed a new FL algorithm called CSFedAvg, where the weight divergence is defined as a utility function to identify the non-IID degree of clients and clients with lower weight divergence are selected with higher priority per round for training the model. However, this approach only considers the accuracy without considering the time cost of the FL training. Lai et al. [23] proposed a client selection framework named Oort to improve the performance of FL training. In Oort, a novel utility of the client is defined by combining the statistical utility (i.e., training loss) with the global system utility (i.e., training time), and clients who can improve both the model accuracy and training speed are selected preferentially. However, the main drawback of Oort is its coarse-grained selection strategy that only considers the data and system heterogeneity between the selected and non-selected clients without considering the differences among the selected clients. Li et al. [24] solved this problem and proposed PyramidFL, which selects clients based on the global utility first and then optimizes the local utility for further client selection. Thus, PyramidFL can prioritize selecting those clients with higher utility in a fine-grained manner. Although the utility-function-based client selection approaches are efficient and lightweight, the utility functions are usually defined by empirical formulas without the support of the theoretical models.

For clarity, a comparison of the related client selection algorithms is presented in Table 1, where the category, solved problem, and disadvantage of these algorithms are summarized. It can be seen from the table that different FL client selection algorithms have their own characteristics and limitations. Choosing the appropriate algorithm should depend on specific application scenarios, resource conditions, and security requirements.

Therefore, in this paper, we adopt a client selection approach based on the system model and optimization problem, whose objective is to maximize the number of selected clients within a specified training time deadline per round. Based on this, we propose a new client selection method based on the genetic algorithm, which can increase the number of clients selected per round compared with the existing heuristic and greedy algorithms. In addition, we consider both system and data heterogeneity among clients to enhance the training performance of FL without compromising the privacy of clients' local data distributions.

3 System model

3.1 Federated learning

In the current Internet, everyone has their own mobile devices and data. If these data are accessible, we can train a high-performance model on them. However, many data involve personal privacy [28], and users are not willing to expose them. Therefore, it is impractical to centralize user data for model training. FL is a decentralized learning framework. In FL, the server first randomly initializes the global model and selects a set of clients to participate in a round of training. Then, the server distributes the parameters of the global model to these clients, who train the model locally with their data and upload the trained model parameters to the server. After that, the server aggregates the updates from multiple clients to obtain a better global model. This process is repeated until the global model reaches the desired performance. The advantage of FL is that clients only need to upload the model parameters without uploading local data. However, the computing and communication capabilities of the clients, as well as the client selection strategy, are crucial to the performance of FL [29].

3.2 System assumption

Similar to related work [15], we consider a MEC platform located in a wireless network, consisting of a server and a set of clients. The server is responsible for distributing and aggregating the global model parameters, as well as coordinating client selection and network resource scheduling. We also employ a Resource Block (RB)-based strategy for network bandwidth allocation. RB is the smallest unit of bandwidth resources defined in LTE [30], and we assume that the number of RBs is limited and managed by the server. Considering the differences in modulation and coding scheme of radio communications among clients, the throughput for each client to upload model parameter is different though the amount of allocated RBs is constant. For simplicity, we assume that the network bandwidth of the i-th client is B_i. Let the size of the global model parameters be S. Therefore, the communication delay of the i-th client can be represented as:

$$t_i = \frac{S}{B_i} \tag{1}$$

Besides, same as the assumption in [15], we also assume that selected clients upload their trained models one by one. Because the amount of allocated RBs is constant, if multiple clients upload their model simultaneously, they will share the RBs. Thus, the communication delay for

Table 1 Comparison of related client selection algorithms

Category		Algorithm	Solved problem		Disadvantage
			System heterogeneity	Data heterogeneity	
Optimization-based	Maximizing client number	FedCS [15]	√	×	• Limitations of greedy algorithms
		Hybrid-FL [16]	√	√	• Limitations of heuristic algorithms • Privacy issues of data sharing
		FedMCCS [17]	√	×	• Limitations of heuristic algorithms
		Xu et al. [18]	√	×	• Difficulties to solve mixed integer programming problems
	Minimizing training time	Shi et al. [19]	√	×	• Challenges to balance learning efficiency and training time
		TRAIN [20]	√	×	• No consider computing capacities of clients
		Wang et al. [21]	√	√	• Challenges to balance performance and energy consumption
		Kang et al. [27]	√	×	• Impractical system model • Large computation overhead
Utility-function-based		CSFedAvg [22]	×	√	• No consider time cost of training
		Oort [23]	√	√	• Coarse grained client selection
		PyramidFL [24]	√	√	• Design of utility function relies on empirical data

multiple clients to upload all models in sequential and parallel mode will be the same.

Furthermore, the computation delay in FL primarily refers to the time spent by clients in updating the model locally. Let c_i be the i-th client's computational capability, which is simply defined by how many data samples it could process per second to update the model. We denote the number of data samples owned by the i-th client as d_i and the number of epochs that the local model is trained locally in each round as e. Therefore, the computation delay for the i-th client to update the model locally in each round can be represented as:

$$\hat{t_i} = \frac{e \cdot d_i}{c_i} \tag{2}$$

3.3 Runtime analysis

We assume that the indices of all participating clients form a set $\mathbf{M} = \{1, 2, \cdots, |\mathbf{M}|\}$. The selected clients in a training round can be denoted as a sequence of $\mathbf{q} = <q_1, q_2, \cdots, q_j, \cdots, q_{|\mathbf{q}|}>$, where $q_j \in \mathbf{M}$. We denote the total communication and computation delay consumed by the first j clients in $\mathbf{q}$ as $\Theta_j^{\mathbf{q}}$ as follows:

$$\Theta_j^{\mathbf{q}} = \begin{cases} 0, & j = 0 \\ T_j^{\mathbf{q}} + \widehat{T}_j^{\mathbf{q}}, & \text{otherwise} \end{cases} \tag{3}$$

where $T_j^{\mathbf{q}}$ and $\widehat{T}_j^{\mathbf{q}}$ are the total communication and computation delay consumed by the first j clients in $\mathbf{q}$, respectively. Since the clients upload models one by one, we have $T_j^{\mathbf{q}} = \sum_{i=1}^{j} t_{q_i}$. Conversely, the clients can update the model while a previous client is in the uploading state, if the computation delay $\hat{t}_{q_j}$ of the j-th client is not exceeded the total communication and computation delay of all previous clients $\Theta_{j-1}^{\mathbf{q}}$, $\widehat{T}_j^{\mathbf{q}}$ will not increase, thus $\widehat{T}_j^{\mathbf{q}} = \sum_{i=1}^{j} \max\{0, \hat{t}_{q_i} - \Theta_{i-1}^{\mathbf{q}}\}$.

3.4 Problem formulation

Our objective in the client selection algorithm is to maximize the number of participating clients in each round within a time deadline. This objective is based on the result in [15], which states that the number of clients participating in training in each round is inversely proportional to the time required to achieve the desired performance of the global model. To achieve this objective, the server needs to select a sequence with as many clients as possible capable of completing the model update and upload within the round's deadline.

Let T represent the deadline for each round, therefore, the optimizing client selection can be formulated as the following maximization problem:

$$\max_{\mathbf{q}}|\mathbf{q}| \quad \text{s.t. } \Theta^{\mathbf{q}}_{|\mathbf{q}|} \leq T \tag{4}$$

Since the global model parameters can be broadcast to all selected clients by the server at once, the latency of the model distribution is the same for all clients. Here, we assume that the server has enough bandwidth for broadcast, thus that the latency of the model distribution can be ignored and will not affect the solution of the above-mentioned optimization problem. Besides, we assume that the server has stronger computational capabilities. Hence, the time consumed for client selection and model aggregation can be negligible as well. For simplicity, in this paper, we assume the system is stable and no crash for any client.

4 Optimization algorithm of client selection

4.1 FedCSGA algorithm

The optimization problem proposed by Eq. (4) is a constrained knapsack problem, which is NP-hard [31]. To address this problem, Nishio et al. [15] proposed the FedCS protocol for client selection in FL based on greedy algorithm [32]. However, the greedy algorithm easily leads to local optima in the optimization process, and this protocol does not consider the data heterogeneity in FL. To overcome these limitations, we propose the FedCSGA algorithm for client selection in FL.

The basic idea of FedCSGA is to define the sequence of selected clients $\mathbf{q}$ as chromosome and employ the genetic algorithm to search for the longest sequence $\mathbf{q}$ that the total consumed time (i.e., communication and computation delay) satisfies the constraint of deadline each round in Eq. (4). This algorithm overcomes the issue of local optima that will be occurred with greedy algorithms thereby leading to the improved convergence and training speed of FL.

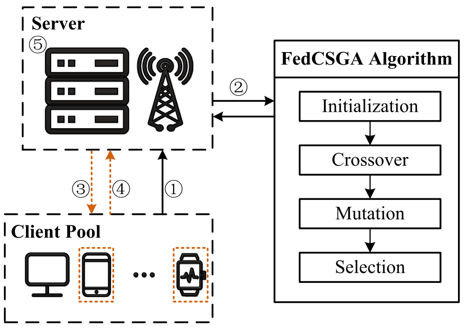


Fig. 1 Framework of FedCSGA. The orange dotted boxes indicate the selected clients participating in the current round. The orange dotted arrows indicate the corresponding steps for the selected clients

We present the framework of the FedCSGA algorithm in Fig. 1, where ① Clients in the client pool upload their resource profiles, which include the communication and computation delay, etc., to the server. ② The server refers to the information in these profiles and uses the FedCSGA algorithm to optimize and select the clients participating in the current round of FL training through a series of operations such as initialization, crossover, mutation, and selection. ③ The server distributes the global model parameters to the selected clients. ④ The selected clients train the model locally and upload new model parameters to the server. ⑤ The server aggregates the models from clients to update the global model and prepares for the next round of training.

The details of FedCSGA are described as follows:

- **Initialization** According to the objective of the maximization problem, in FedCSGA, the chromosome is defined as the sequence of the selected clients $\mathbf{q}$, which considers not only the presence or absence of a client to participate in training but also the order of the selected clients. Let n be the size of population, and $\mathbf{P}_r = <\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_i, \cdots, \mathbf{q}_n>$ represent the r-th generation population. Here, $\mathbf{q}_i = <q_{i1}, q_{i2}, \cdots, q_{ij}, \cdots, q_{i|\mathbf{q}_i|}>$ denotes the i-th chromosome of the population, where q_{ij} is the j-th index of clients (i.e., gene) in chromosome $\mathbf{q}_i$. To increase the diversity of the population, the chromosome $\mathbf{q}_i$ is initialized as follows: Firstly, a client is selected randomly from $\mathbf{M}$ as the first gene q_{i1}. Then, a greedy method is used to add a new client that consumes the least total communication and computation delay $\Theta^{\mathbf{q}_i}_{|\mathbf{q}_i|}$ to $\mathbf{q}_i$ iteratively until $\Theta^{\mathbf{q}_i}_{|\mathbf{q}_i|}$ reaches the deadline for each round T. The above process is repeated n times to form the initial population.
- **Fitness function** Considering the problem given in Eq. (4) is a constrained optimization problem, we define the fitness function $F(\mathbf{q}_i)$ with a penalty for the i-th chromosome as follows:

$$F(\mathbf{q}_i) = h(\mathbf{q}_i) - \lambda g(\mathbf{q}_i) \tag{5}$$

where $h(\mathbf{q}_i)$ and $g(\mathbf{q}_i)$ represent the objective and the penalty functions for the i-th chromosome, respectively, and $\lambda \in [0, 1)$ is the adjustment factor for the penalty function. Since the objective of Eq. (4) is to maximize the number of clients in $\mathbf{q}_i$, we define $h(\mathbf{q}_i)$ as follows:

$$h(\mathbf{q}_i) = |\mathbf{q}_i| \tag{6}$$

Furthermore, due to the constraint is the deadline for each round T, we define the penalty function $g(\mathbf{q}_i)$ as follows:

$$g(\mathbf{q}_i) = e^{\max\left(0, \frac{\Theta^{\mathbf{q_i}}_{|\mathbf{q_i}|} - T}{T}\right)} - 1 \tag{7}$$

According to Eq. (6), when $\Theta^{\mathbf{q_i}}_{|\mathbf{q_i}|}$ satisfies the constraint condition in Eq. (4), $g(\mathbf{q}_i) = 0$ and thus there is no penalty on the fitness function $F(\mathbf{q}_i)$. Otherwise, when $\Theta^{\mathbf{q_i}}_{|\mathbf{q_i}|} > T$, value of the penalty function will increase significantly with the increasing of $\Theta^{\mathbf{q_i}}_{|\mathbf{q_i}|}$. In addition, we define the adjustment factor λ for the penalty function as

$$\lambda = \lambda_0 e^{\sqrt{r}} \tag{8}$$

where r is the number of generation and λ_0 is the penalty factor with a value of 0.8. According to Eq. (8), λ increase with the increasing of r. That is to say, at the beginning of the evolution of the genetic algorithm, chromosomes will receive a smaller penalty if they do not meet the constraint. This is beneficial for the algorithm to search for more sequences of the selected clients. Then, chromosomes unsatisfied with constraint will receive greater and greater penalties with the process of evolution, which ensures the genetic algorithm to obtain the optimal and feasible chromosomes at the end of the evolution.

- **Crossover operator** We adopt Uniform Crossover (UX) [33], where two parents are selected randomly from the current population, and then each gene position of the parents is exchanged with a crossover probability p_c to generate new offspring. Compared with single-point and two-point crossover, UX allows each gene locus to have the potential to be crossed over. Besides, to avoid premature convergence and maintain population diversity, we employ an approach of adaptive crossover probability [34], where p_c is varied depending on the fitness values of the parents involved in the crossover operation. Suppose $\mathbf{q}_i$ and $\mathbf{q}_j$ are two chromosomes selected for crossover, then the crossover probability p_c is defined as follows:

$$p_c = \begin{cases} \dfrac{k_1(F_{max} - F_{larger})}{F_{max} - F_{avg}} & , F_{larger} \geq F_{avg} \\ k_2 & , F_{larger} < F_{avg} \end{cases} \tag{9}$$

where $F_{max} = \max_{i \in [1,n]} F(\mathbf{q}_i)$ and $F_{avg} = \frac{1}{n}\sum_{i=1}^{n} F(\mathbf{q}_i)$ represent the maximum and average fitness values in the population, respectively. And, $F_{larger} = \max\{F(\mathbf{q}_i), F(\mathbf{q}_j)\}$ represents the larger fitness value between chromosomes $\mathbf{q}_i$ and $\mathbf{q}_j$, k_1, k_2 are constants, and usually $k_1 < k_2$. The characteristic of the adaptive crossover probability is that when the fitness values of the chromosomes in the population tend to be consistent or reach a local optimum, it increases the crossover probability. Conversely, when the fitness values of the population are more dispersed, it decreases the crossover probability. At the same time, chromosomes with fitness values higher than the average fitness value of the population will be assigned lower crossover probabilities, which will help them enter the next generation. On the other hand, chromosomes with fitness values lower than the average fitness value will have higher crossover probabilities, which will lead to their elimination. In addition, considering the different gene lengths of the chromosomes in the population, the crossover operation should be performed at the gene position of two parents one by one until reaching the end of the shorter parent. Moreover, considering the uniqueness of the gene (i.e., the index of the client) in a chromosome, if the crossover operation causes a duplicate gene at a gene position, it must be aborted at this position and skipped to the next position. Here is an example of UX. Suppose two selected parent chromosomes are $\mathbf{q}_1 = <1, 2, 3>$ and $\mathbf{q}_2 = <2, 4, 5, 6>$, which are shown in Fig. 2a. Assuming $F_{max} = 5.0$, $F_{avg} = 4.0$, $F(\mathbf{q}_1) = 3.0$, and $F(\mathbf{q}_2) = 4.0$, we calculate the crossover probability according to Eq. (9) and obtain $p_c = 0.5$. To perform UX, we generate a random number $\rho \in [0, 1)$ at each gene position. If $\rho < p_c$, the UX will be performed at this position. Otherwise, this position will be skipped. Thus, the process of the UX is shown as follows: At the first gene position, we have $\rho = 0.3 < p_c$, and crossover should be performed here. However, exchanging $q_{11} = 1$ and $q_{21} = 2$ would result in the duplicate gene $<2>$ in $\mathbf{q}_1$, thus this position is skipped. Next, at the second gene position, we have $\rho = 0.6 > p_c$ and do not perform crossover at this position. Then, at the third gene position, $\rho = 0.4 < p_c$, we exchange the genes of $\mathbf{q}_1$ and $\mathbf{q}_2$ at this position. Since the current position is the last one of the shorter parent $\mathbf{q}_1$, we obtain two new offspring chromosomes $\mathbf{q}'_1 = <1, 2, 5>$ and $\mathbf{q}'_2 = <2, 4, 3, 6>$ shown in Fig. 2b.

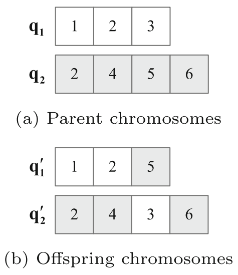


Fig. 2 An example of UX: **a** Parent chromosomes **b** Offspring chromosomes

- **Mutation operator** The traditional Swap Mutation (SM) [35] introduces new genetic information into the population by swapping two gene positions in a chromosome. However, in this paper, if the exchanged genes are widely separated in the chromosome, it may lead to a sharp decrease in the chromosome's fitness value. Therefore, we introduce a new mutation method called Adjacent Swap Mutation (ASM), where each gene position is limited to only exchange with its adjacent next gene in the chromosome with a mutation probability p_m. In particular, when the mutation occurs in the last gene position of the chromosome, we employ the same greedy method as in the initialization to append a new client that consumes the lowest total communication and computation delay to the chromosome. Besides, to achieve a better balance between exploration and exploitation, we also adopt an adaptive mutation probability [34], where p_m is dynamically adjusted based on the fitness value of the chromosome in the progress of evolution. Similar to the crossover operation, the mutation probability p_m of chromosome $\mathbf{q}_i$ is defined as follows:

$$p_m = \begin{cases} \dfrac{k_3(F_{max} - F(\mathbf{q}_i))}{F_{max} - F_{avg}} & , F(\mathbf{q}_i) \geq F_{avg} \\ k_4 & , F(\mathbf{q}_i) < F_{avg} \end{cases} \tag{10}$$

where k_3 and k_4 are constants, and usually $k_3 < k_4$. Here is an example of ASM. As shown in Fig. 3a, there is a chromosome $\mathbf{q}_1 = <1, 2, 3>$. Assuming $F_{max} = 5.0$, $F_{avg} = 4.0$, and $F(\mathbf{q}_1) = 3.0$, we calculate the mutation probability according to Eq. (10) and obtain $p_m = 0.04$. Similar to the crossover operation, the process of ASM is as follows: At the first gene position, we generate a random number $\rho = 0.03$. Since $\rho < p_m$, the mutation should be performed, and the first gene $<1>$ is exchanged with its adjacent gene $<2>$. Next, at the second gene position, we have $\rho > p_m$, the mutation is not operated. Then, at the last gene position, we have $\rho < p_m$. Thus, we append a new client k that consumes the least total communication and computation delay to $\mathbf{q}_1$, that is, $k = \arg\min_{k \in \mathbf{M} \wedge \mathbf{k} \notin \mathbf{q}_1} \Theta^{\mathbf{q}_1 + <\mathbf{k}>}_{|\mathbf{q}_1 + <\mathbf{k}>|}$. Suppose $k = 4$, as shown in Fig. 3b, $\mathbf{q}_1' = <2, 1, 3, 4>$ is the newborn chromosome after ASM.

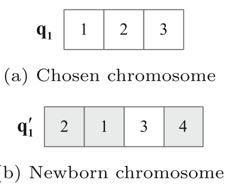


Fig. 3 An example of ASM: **a** Chosen chromosome **b** Newborn chromosome

- **Selection operator** The selection operator is a process of survival of the fittest based on fitness values. We employ the binary tournament method to select parent chromosomes [26]. In each iteration, we randomly select two chromosomes from the population and choose the chromosome with the higher fitness value to enter the next generation population. We repeat this process until the size of the new population reaches the original population size n.

Algorithm 1 FedCSGA Algorithm

```
Input:
    $\mathbf{M}$: set of indices of all participating clients
    $\hat{t}_i$: computation delay of the $i$-th client
    $t_i$: communication delay of the $i$-th client
    $T$: deadline of each round
    $n$: size of population
    $R$: total number of evolutionary generations
Output:
    $\mathbf{q}^*$: the best chromosome
1:  $r \leftarrow 1$
    /*Initialization*/
2:  for $i = 1$ to $n$ do
3:      $\mathbf{q}_i \leftarrow \varnothing$
4:      $k \leftarrow$ randomly select a client $k \in \mathbf{M}$ with $\Theta_1^{\langle k \rangle} < T$
5:      while $\Theta_{|\mathbf{q}_i + \langle k \rangle|}^{\mathbf{q}_i + \langle k \rangle} < T$ do
6:          $\mathbf{q}_i \leftarrow \mathbf{q}_i + \langle k \rangle$
7:          $k \leftarrow \underset{k \in \mathbf{M} \wedge k \notin \mathbf{q}_i}{\arg\min}\ \Theta_{|\mathbf{q}_i + \langle k \rangle|}^{\mathbf{q}_i + \langle k \rangle}$
8:      end while
9:      add $\mathbf{q}_i$ to $\mathbf{P}_r$
10: end for
11: while $r < R$ do
        /*Crossover*/
12:     for $c = 1$ to $\lceil n/2 \rceil$ do
13:         randomly select two chromosomes $\mathbf{q}_i$ and $\mathbf{q}_j$ from $\mathbf{P}_r$
14:         $\mathbf{q}_i', \mathbf{q}_j' \leftarrow \mathrm{UX}(\mathbf{q}_i, \mathbf{q}_j)$ with adaptive $p_c$
15:     end for
        /*Mutation*/
16:     for $c = 1$ to $n$ do
17:         randomly select a chromosome $\mathbf{q}_i$ from $\mathbf{P}_r$
18:         $\mathbf{q}_i' \leftarrow \mathrm{ASM}(\mathbf{q}_i)$ with adaptive $p_m$
19:     end for
        /*Selection*/
20:     for $c = 1$ to $n$ do
21:         randomly select two chromosomes $\mathbf{q}_i$ and $\mathbf{q}_j$ from $\mathbf{P}_r$
22:         $\widehat{\mathbf{q}} \leftarrow \underset{\mathbf{q} \in \{\mathbf{q}_i, \mathbf{q}_j\}}{\arg\max}\ F(\mathbf{q})$
23:         add $\widehat{\mathbf{q}}$ to $\mathbf{P}_{r+1}$
24:     end for
25:     $r \leftarrow r + 1$
26: end while
27: $\mathbf{q}^* \leftarrow \underset{\mathbf{q} \in \mathbf{P}_r \wedge \Theta_{|\mathbf{q}|}^{\mathbf{q}} \leq T}{\arg\max}\ F(\mathbf{q})$
28: return $\mathbf{q}^*$
```

Algorithm 1 shows the pseudo-code of FedCSGA. At first, n chromosomes are generated to initialize the population (lines 2–10). In each iteration of evolution, the UX crossover (lines 12–15) and ASM mutation (lines 16–19) operators are performed. After that, a new generation population with n chromosomes is constructed using the selection operator (lines 20–24). After R evolutionary generations, we select the best chromosome $\mathbf{q}^*$, which is the longest chromosome satisfying the constraint condition in Eq. (4) (lines 27–28).

Next, we give a time complexity analysis of Algorithm 1. The time complexity of the initialization is $O(n|\mathbf{M}|)$. The computing time of a UX operation depends on the length of chromosomes, whose maximum value is the length of the best chromosome $\mathbf{q}^*$. Thus, the time complexity of a UX operation is $O(|\mathbf{q}^*|)$, and the time complexity of the crossover operation each generation is $O(n|\mathbf{q}^*|)$. Since there is a greedy method in the ASM operation, the time complexity of an ASM operation is $O(|\mathbf{q}^*| + |\mathbf{M}|)$, and the time complexity of the mutation operation each generation is $O(n(|\mathbf{q}^*| + |\mathbf{M}|))$. The time complexity of the selection operation is $O(n)$. Considering the total generation is R and $|\mathbf{q}^*| < < |\mathbf{M}|$, the time complexity of Algorithm 1 is $O(nR|\mathbf{M}|)$. Compared to the computation and communication of clients in FL, the time consumed by FedCSGA is very small and can be negligible. For example, in our experiments, we set $n = 90$, $R = 10$, and $|\mathbf{M}| = 100$, the once running time of FedCSGA is only 0.11 s, while the deadline for each round T is about 3–5 min. Besides, FedCSGA also does not have excessive communication costs. To run FedCSGA, the server only needs clients to transmit their computation and communication delay. These parameters are all numerical values, which communication overhead can be ignored compared with that of the parameters of the training model.

The convergence of genetic algorithms cannot be guaranteed, but there are some strategies (e.g., proper parameters, evolutionary operations, population diversity, etc.) to improve its convergence. In this paper, we evaluate the convergence of FedCSGA by a sensitivity analysis of the genetic algorithm parameters (e.g., population size, crossover and mutation probabilities, etc.) in experiments. The theoretical convergence analysis will be left in future work. Besides, the computation and communication capacities of clients may change during FL training in practice. To deal with this issue, we can use the weighted smoothing method to estimate the current computation and communication delay of clients. More sophisticated methods are out of the scope of this paper and will be studied in future work as well.

4.2 Consideration of non-IID data

As aforementioned, the non-IID data is a critical problem for FL. In practical scenarios, the usage patterns of specific client's mobile devices determine the training data on each client. Consequently, there can be significant variations in the distribution of local datasets among clients. Because of this, the non-IID data will lead to weight discrepancies in local models, that is, local models with the same initial weight parameters will converge to different models. This results in a slower convergence and lower accuracy of the model in FL.

To this end, inspired by the related work on non-IID data [36–38], we introduce the model accuracy into the client selection of FL. In each FL round, clients upload both their model parameters and accuracy after local training, and then the server selects clients participating in the next round of training based on the total consumed time and model accuracy of clients. Specifically, we redefine the objective function $h(\mathbf{q}_i)$ of fitness function in Eq. (5) as follows:

$$h(\mathbf{q}_i) = \sum_{j=1}^{|\mathbf{q_i}|} (1 - \alpha A(q_{ij})) \tag{11}$$

where $A(q_{ij})$ represents the model accuracy of the j-th client in $\mathbf{q}_i$. $\alpha \in [0, 1]$ is a tunable factor.

According to Eq. (11), we prioritize clients with lower model accuracy to participate in the training. Because a lower accuracy typically indicates that this client's model has been trained with fewer rounds, this client should contribute more to the global model in the next training rounds. Besides, the factor α is used to adjust the importance of the model accuracy in client selection. When $\alpha = 0$, Eq. (11) reduces to the IID case, i.e., Eq. (6), which maximizes the number of clients selected. When $\alpha > 0$, the model accuracy comes into play, and set a proper value of α, we can have a tradeoff between the number of clients and model accuracy in client selection of FL.

It should be noted that there are alternative methods for data heterogeneity (e.g., model parameter deviation, gradient variance, and weighted average, etc.), in this paper, we only introduce the model accuracy of clients into the fitness function. This method is very simple, and the model accuracy is also a numerical value, thus its communication overhead can be ignored. We are going to study other methods in future work.

5 Performance evaluation

5.1 Experiment setup

We implement a discrete-event simulator with Python and Pytorch to evaluate FedCSGA and related FL algorithms. In the simulator, the main procedures of FL are defined as events including client selection, local model training, local model upload, global model aggregation, etc. All events are maintained and processed with an event queue ordered by the time of event occurrence or completion. The system heterogeneity of real-world FL is captured by defining different attributes of the server and clients to events such as the model parameters, network bandwidth, computation capacity, data distribution, etc. Besides, another upload queue is implemented to simulate the sequential communication mode in our system. At the beginning of the simulation, an event-driven engine is started to process a client selection event and invoke its corresponding activities (e.g., client selection and model broadcast by the server), and then a set of local training events of the selected clients are added to the event queue ordered by the computation delay of training. For each of the local training events, the local model updating is performed and the updated model is added to the upload queue. Afterward, the engine can calculate the communication delay for each of the upload elements and generate the event of local model upload. All local model upload events will be collected and delivered to the server for global model aggregation. After the aggregation, a new client selection event is generated for the next training round. In this way, the procedure of the FL system and algorithm can be simulated only with a single process.

All experiments are executed on a server equipped with an NVIDIA RTX-3080 GPU and an Intel i7-10700 CPU. We configure an MEC environment implemented in a wireless network, which consists of a server and 100 clients.

5.1.1 Datasets

Our training tasks are based on two well-known public datasets. The first dataset is Fashion-MNIST [39], which contains 10 classes of fashion products, consisting of 60,000 training images and 10,000 testing images. The other dataset is CIFAR-10 [40], a classic object classification dataset consisting of 50,000 training images and 10,000 testing images with 10 object classes.

5.1.2 Models

The training tasks for two datasets are performed by a CNN model. It consists of six 3×3 convolution layers with the number of channels of 32, 32, 64, 64, 128, and 128, respectively. Each channel is applied ReLU and is batch normalized, and each two channels is followed by a 2×2 maximum pool layer to reduce the feature space. The network is followed by three fully-connected layers. The first two layers contain 382 and 192 units respectively, which are activated by ReLU. The last layer contains 10 units, and the softmax is applied to output the classification results. In addition, because the Fashion-MNIST dataset contains 28×28 pixels of gray images, while the CIFAR-10 dataset contains 32×32 pixels of color images, the size of the global model parameters is different between the two tasks. Specifically, the model for the Fashion-MNIST dataset involves about 3.6 million parameters (i.e., $S =$ 14.4 Mbytes), while the model for the CIFAR-10 dataset involves about 4.6 million parameters (i.e., $S = 18.3$ Mbytes).

The objective of this paper is to optimize the client selection within a time deadline of each round not the specific training model and task itself. Hence, we choose the relatively simple CNN model with two well-known public Fashion-MNIST and CIFAR-10 datasets as the same as FedCS [15] and Hybrid-FL [16]. We think this choice can sufficiently represent the challenges in FL by setting different system and algorithm parameters in simulation, and the experimental results are valid for other models and datasets in practical systems.

5.1.3 Heterogeneous setup

In experiments, we randomly distribute the image data of the two training datasets to all clients and make the number of data samples owned by the i-th client (i.e.,d_i) in a range of 100 to 1,000. Here, we configure two data distribution settings for training as follows:

- IID setting: each client randomly selects the specified number of data samples from the whole training dataset;
- Non-IID setting: the classes of image data each client holds follow a truncated normal distribution $\psi(2, 0.7, 0.5, 10.5)$, which denotes the mean and variance of the number of the data classes owned by each client are 2 and 0.7 with the lower and upper bounds as 0.5 and 10.5. This ensures that the number of classes owned by each client ranges from 1 to 10 classes inclusive. Each client then sampled the specified number of images randomly from different subsets.

5.1.4 Parameters setup

(1) Training hyperparameters When updating the model, we selected the following hyperparameters: 50 for mini-batch size, 5 for the number of epochs that the local model is trained locally in each round (i.e., e), 0.25 for the initial learning rate of SGD updates, and 0.99 for learning-rate decay.

(2) System and algorithm parameters Wireless communications are modelled based on LTE networks. We assume that the network bandwidth of the i-th client B_i conforms to a truncated normal distribution $\psi(1.4, 2.7, 0, 8.6)$, which denotes the mean and variance of B_i are 1.4 Mbit/s and 2.7 Mbit/s with the lower and upper bounds as 0 and 8.6 Mbit/s.

Meanwhile, we determined the computational capability of the i-th client c_i randomly from a range of 10 to 100 /s. As a result, in each round, the computation time t_i for the i-th client to update the model locally varied from 5 to 500 s average.

In FedCSGA, we set the size of population n to 90 and the total number of evolutionary generations R to 10. Besides, let the crossover probability constants $k_1 = 0.5$ and $k_2 = 0.9$, the mutation probability constants $k_3 = 0.02$ and $k_4 = 0.05$. Each experiment is run for 10 times, and we take the average of the results.

5.1.5 Baselines

We evaluate the performance of FedCSGA compared with several representative and related baselines of FL algorithms including FedAvg [5], FedCS [15], Hybrid-FL [16], Oort [23] and PyramidFL [24] as follows:

- FedAvg: the first FL algorithm that selects clients randomly to participate in training per round until the deadline of each round.
- FedCS: a FL protocol that selects clients with a greedy algorithm that makes the server aggregate as many client updates as possible within the time deadline for each round.
- Hybrid-FL: a hybrid FL mechanism that selects clients considering both system and data heterogeneity.
- Oort: a FL framework that selects clients to participate in training per round based on a client utility metric combining both statistical and system efficiency.
- PyramidFL: a fine-grained FL algorithm that selects clients based on both global utility and local utility.

Since FedCSGA is based on the optimization problem and sensitive to the deadline for each round, we compare it with the above-mentioned baselines under the best-chosen T to show the advantages of the optimization methods in this paper.

5.1.6 Performance metrics

We evaluate different FL algorithms with the following metrics:

- Accuracy: the model accuracy on the test datasets in training.
- Number of selected clients: including the number of clients selected in each round and the total number of selected clients.

5.2 Evaluation results

5.2.1 Comparison with baselines

(1) IID setting

Figure 4 shows the number of clients selected in each round of different FL algorithms on two datasets with the IID setting and $T = 3$ min. From Fig. 4a, which is on the Fashion-MNIST dataset, we can find that the number of clients selected by FedAvg in each round is much less than that of other algorithms. This is because FedAvg only selects clients for training randomly. FedCS focuses on the system heterogeneity of clients and greedily selects clients with the shortest training and upload time in each round, and then effectively increases the number of clients selected in each round. Besides the aforementioned time, Hybrid-FL further considers the statistical deviation of different classes of data used for model updating. Although the data distribution of clients should be uniform in the IID setting, there will still be a small discrepancy among different clients in the coefficient of variation, which will affect the selection of those clients with shorter time, and thus the number of clients selected in each round of Hybrid-FL is less than that of FedCS. Similarly, Oort and PyramidFL consider both data heterogeneity to improve the model accuracy and system heterogeneity to reduce the time required to perform the training task. And, the number of clients selected in each round of Oort, PyramidFL, and Hybird-FL is approximate. At last, FedCSGA searches for the relatively optimal solution through a genetic algorithm when selecting clients, and uses an optimization greedy algorithm for fine-tuning, which effectively alleviates the problem of FedCS and further improves the the number of clients selected in each round. The results on the CIFAR-10 dataset are presented in Fig. 4b, which are similar to those observed on the Fashion-MNIST dataset.

Figure 5 shows the accuracy curves of different FL algorithms on two datasets with the IID setting and $T = 3$

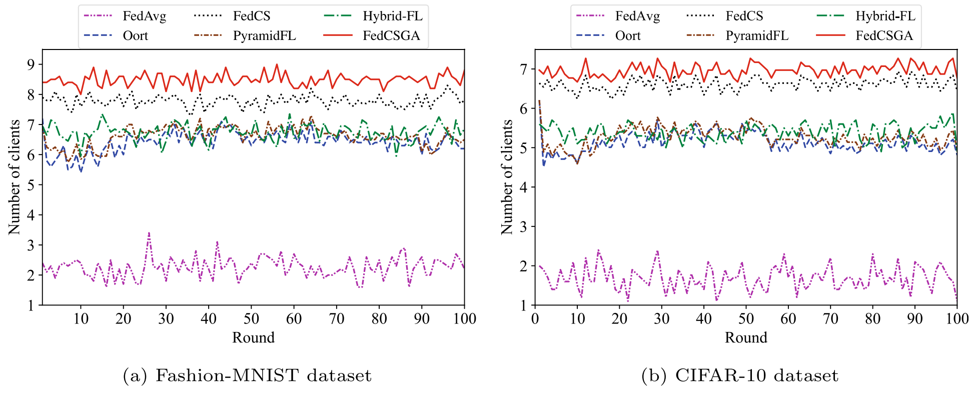


Fig. 4 Number of clients selected in each round of different FL algorithms on two datasets with IID setting and $T = 3$ min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

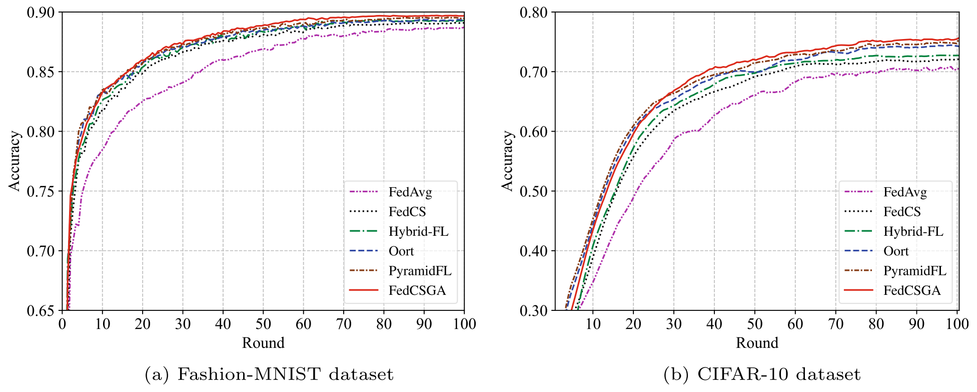


Fig. 5 Accuracy curves of different FL algorithms on two datasets with IID setting and $T = 3$ min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

min. In Fig. 5a, we can find that FedAvg has the lowest convergence speed and model accuracy because the number of clients selected in each round is too small. FedCS considers the system heterogeneity of clients and uses a greedy algorithm to select as many clients as possible in each round, so its performance is greatly improved compared to FedAvg. Although the number of clients selected in each round is lower than that of FedCS, Hybrid-FL adjusts the priority of client selection by evaluating the deviation degree of each class of data in the overall data used for model training, and thus the convergence speed and model accuracy of Hybrid-FL are slightly higher than those of FedCS. Moreover, Oort also considers the statistical utility of improving the accuracy of the model and the system utility of the time required to perform the training task. And, PyramidFL selects clients based on both global and local utilities. Therefore, the performance of Oort and PyramidFL is better than Hybrid-FL. Among all these algorithms, FedCSGA has the best model training performance. This is because FedCSGA uses a genetic algorithm to maximize the number of clients selected in each round, thereby training the model more efficiently. The results on the CIFAR-10 dataset are presented in Fig. 5b, which are similar to those observed on the Fashion-MNIST dataset.

For clarity, we show the average number of clients selected in each round and the model convergence accuracy of different FL algorithms on Fashion-MNIST and CIFAR-10 datasets with the IID setting in Table 2. It can be obtained from the table that the number of clients selected per round and the model accuracy of FedCSGA

Table 2 Average number of clients selected in each round and model convergence accuracy (%) of different FL algorithms on two datasets with IID setting and $T = 3$ min (The best result is shown in bold)

Algorithm	Number of clients		Accuracy (%)	
	Fashion-MNIST	CIFAR-10	Fashion-MNIST	CIFAR-10
FedAvg	2.3	1.6	88.7	70.6
FedCS	7.8	6.5	89.1	72.3
Hybrid-FL	6.6	5.3	89.3	72.9
Oort	6.4	5.1	89.3	74.3
PyramidFL	6.5	5.2	89.4	75.1
FedCSGA	**8.5**	**7.0**	**89.9**	**76.0**

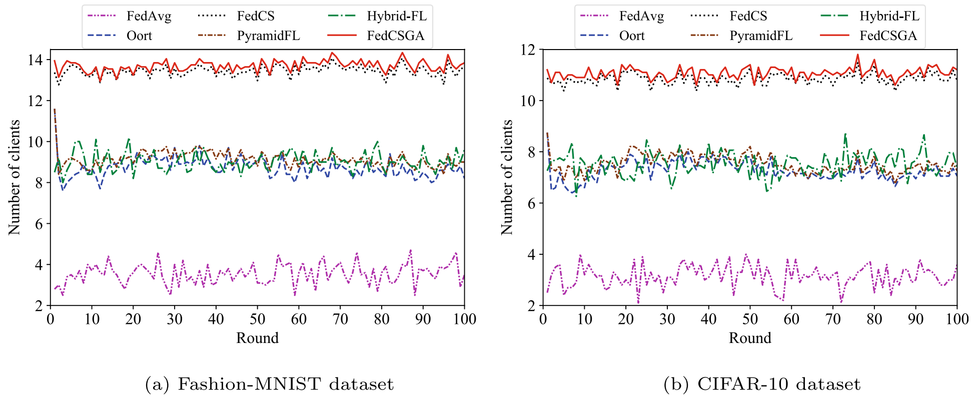


Fig. 6 Number of clients selected in each round of different FL algorithms on two datasets with non-IID setting and $T = 5$ min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

increase by 45.6% and 2.4% on average respectively compared with these baseline algorithms.

(2) Non-IID setting

Figure 6 shows the number of clients selected in each round of different FL algorithms on two datasets with the non-IID setting and $T = 5$ min. In Fig. 6a, as the same as the IID setting, the number of clients selected by FedAvg in each round is the smallest compared with other algorithms, and FedCS can select many more clients in each round than FedAvg. In addition, the number of selected clients of Hybrid-FL, Oort, and PyramidFL is also similar. For FedCSGA, we set the value of α to 0.7. Different from the IID setting, although using a genetic algorithm, FedCSGA just can select as many clients as FedCS in the non-IID setting. This is because the fitness function in Eq. (11) tries to balance the number of selected clients and the model accuracy of clients, while FedCS just greedily selects the fastest clients in each round. The results on the CIFAR-10 dataset shown in Fig. 6b are similar to those on the Fashion-MNIST dataset.

Figure 7 shows the accuracy curves of different FL algorithms on two datasets with the non-IID setting and $T = 5$ min. According to Fig. 7a, the performance of FedCS is better than that of FedAvg. Moreover, Hybrid-FL, Oort, and PyramidFL perform significantly better than FedCS since they consider both system and data heterogeneity. As the same as aforementioned, the performance of FedCSGA is better than that of other algorithms, since FedCSGA can make a better tradeoff between the number of selected clients and model accuracy in the non-IID setting. As shown in Fig. 7b, the performance on the CIFAR-10 dataset is similar to that on the Fashion-MNIST dataset.

Table 3 shows the average number of clients selected in each round and the model convergence accuracy of different FL algorithms on Fashion-MNIST and CIFAR-10 datasets with the non-IID setting. We can obtain from the table that the number of clients selected per round and the model accuracy of FedCSGA increase by 54.3% and 7.7% on average respectively compared with these baseline algorithms.

Therefore, taking into account the results such as the number of selected clients and model accuracy, FedCSGA shows the best performance than the above-mentioned state-of-the-art FL algorithms in both IID and non-IID settings, and the effectiveness of the proposed approach is verified. It should be noted again that the proposed algorithm is practical and can be effectively extended to more complex tasks by optimizing the client sequence to participate in FL training. For the given number of training rounds, FedCSGA can achieve higher model accuracy. For

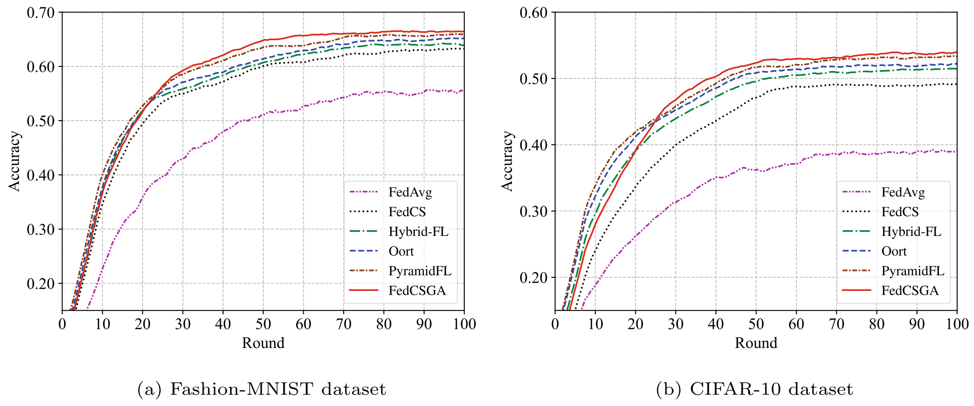


Fig. 7 Accuracy curves of different FL algorithms on two datasets with non-IID setting and $T = 5$ min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

Table 3 Average number of clients selected in each round and model convergence accuracy (%) of different FL algorithms on two datasets with non-IID setting and $T = 5$ min (The best result is shown in bold)

Algorithm	Number of clients		Accuracy (%)	
	Fashion-MNIST	CIFAR-10	Fashion-MNIST	CIFAR-10
FedAvg	3.0	2.9	57.8	38.8
FedCS	13.4	11.1	63.5	49.1
Hybrid-FL	9.1	7.6	63.9	51.2
Oort	8.9	7.5	65.2	52.2
PyramidFL	9.0	7.5	66.1	53.3
FedCSGA	**13.5**	**11.2**	**66.5**	**54.0**

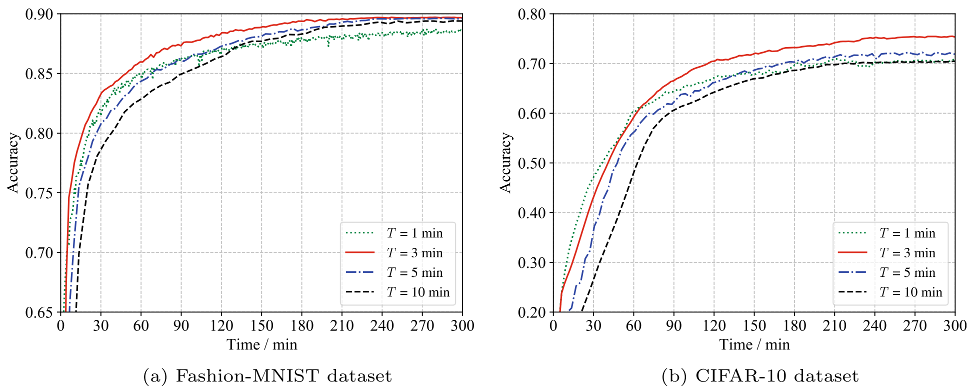


Fig. 8 Accuracy curves of different T on two datasets with IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

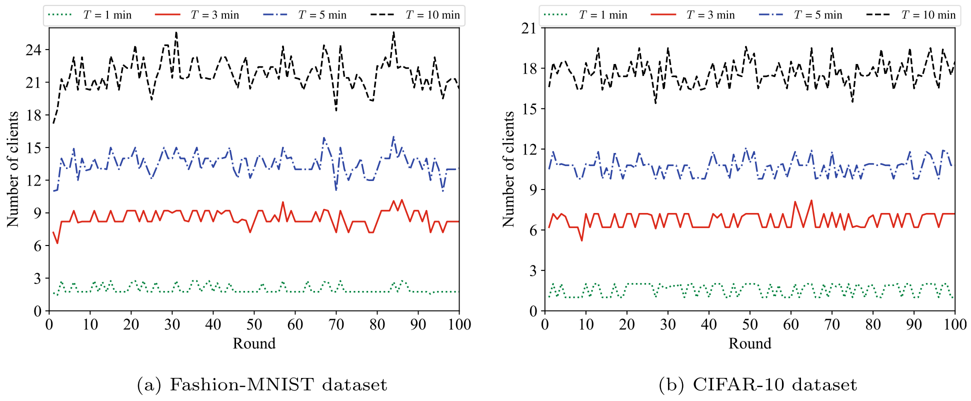


Fig. 9 Number of clients selected in each round of different T on two datasets with IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

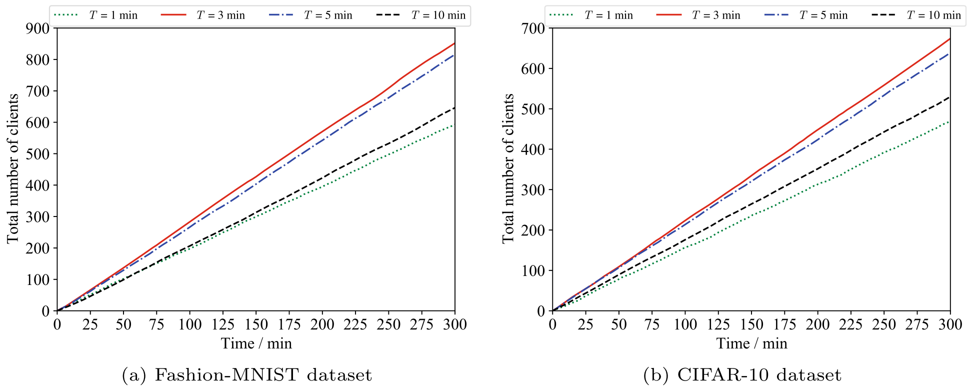


Fig. 10 Total number of selected clients of different T on two datasets with IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

the given target model accuracy, FedCSGA can decrease the training round and time, thus improving resource utilization and training performance effectively.

5.2.2 Impacts of the deadline for each round

(1) IID setting

Figure 8 shows the accuracy curves of different deadline for each round (i.e., T) on two datasets with the IID setting. From Fig. 8a, we observe that the final model accuracy is relatively low when T is too small (e.g., $T = 1$ min) or too large (e.g., $T = 5$ min and $T = 10$ min). And, when T is appropriate (e.g., $T = 3$ min), the performance of FedCSGA can achieve the best. Figure 8b shows the results on the CIFAR-10 dataset, which are similar to those on the Fashion-MNIST dataset.

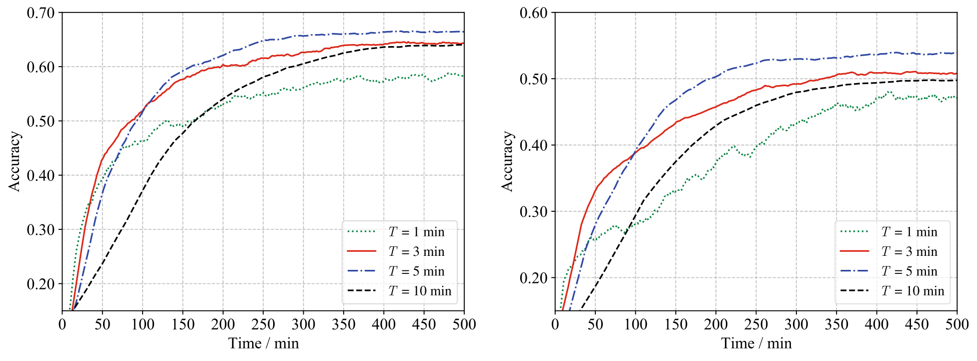
To reveal the reason for the aforementioned observations, we show the number of selected clients of different T on two datasets with the IID setting in Fig. 9 and Fig. 10. In Fig. 9a, we observe that the number of clients selected in each round gradually increases as T increases from 1 min to 10 min. However, it can be seen from Fig. 10a that the total number of selected clients is not strictly proportional to T. When $T = 1$ min, although the aggregation frequency is high, the total number of selected clients is still the lowest since the number of clients selected in each round is very low. Thus, the final accuracy for $T = 1$ min is not high in Fig. 8a. Interestingly, the total number of selected clients when $T = 5$ min and 10 min is less than that of $T = 3$ min, too. This is because although there are more clients selected in each round when T is larger, the aggregation frequency decreases more significantly, which affects the model accuracy in Fig. 8a. Moreover, it can be seen that the best performance in the total number of selected clients and the model accuracy can be obtained when $T = 3$ min. As shown in Figs. 9b, 10b, the performance of the CIFAR-10 dataset is similar to that of the Fashion-MNIST dataset.

(2) Non-IID setting

Figures 11, 12, 13 show the accuracy curves and the number of selected clients of different T on two datasets with the non-IID setting, respectively. Similar to the IID setting, we can observe from Fig. 11 that the best accuracy of FedCSGA can be obtained when T is appropriate (e.g., $T = 5$ min). However, it should be noted that the model training performance depends not only on the number of selected clients but also on the data distribution of the selected clients in the non-IID setting, which is different from the IID setting. Although the total number of selected clients when $T = 5$ min is slightly less than that of $T = 3$ min shown in Fig. 13, the number of clients selected in each round when $T = 5$ min shown in Fig. 12 is more than that of $T = 3$ min, and more clients are conducive to alleviate the non-IID degree of data and improve the training effect of each round. Therefore, under the non-IID setting, the best accuracy of FedCSGA is achieved when $T = 5$ min.

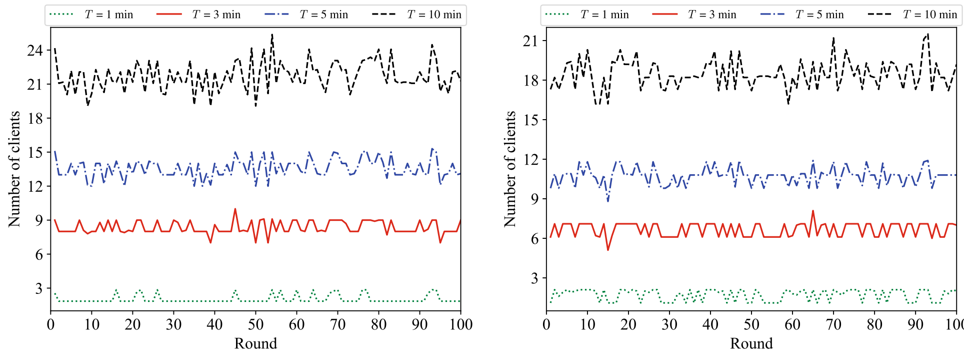
5.2.3 Impacts of the factor for non-IID setting

Figure 14 shows the accuracy curves of different factor (i.e., α) on two datasets with the non-IID setting and $T = 5$ min. As shown in Fig. 14a, the final model accuracy increases as α increases from 0 to 0.7, but decreases as α increases to 1. This is because factor α is used to adjust the optimization objective of FedCSGA according to Eq. (11). When $\alpha = 0$, Eq. (11) reduces to the IID case, where FedCSGA only tries to maximize the number of selected clients but does not consider the model accuracy in each round. When $\alpha = 1$, the weight of the model accuracy in Eq. (11) reaches the maximum, which will affect negatively the number of clients selected in each round. When α has an appropriate value (i.e., $\alpha = 0.7$), FedCSGA achieves a balanced tradeoff between the number of selected clients and model accuracy, and thus the performance of the model accuracy is best in the FL training. Therefore, the effectiveness of the factor α is verified in the non-IID setting. As shown in Fig. 14b, the results on the CIFAR-10 dataset are similar to that on the Fashion-MNIST dataset.



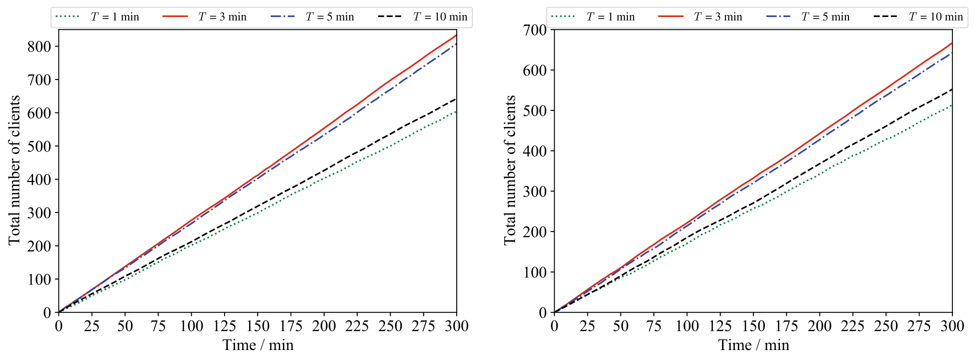
(a) Fashion-MNIST dataset (b) CIFAR-10 dataset

Fig. 11 Accuracy curves of different T on two datasets with non-IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset



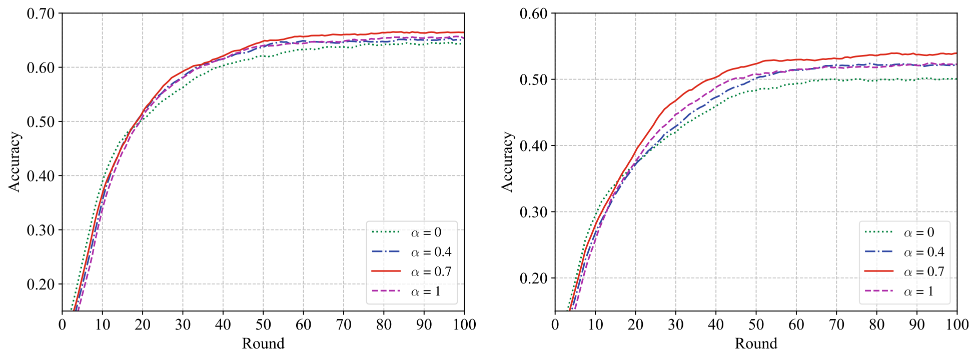
(a) Fashion-MNIST dataset (b) CIFAR-10 dataset

Fig. 12 Number of clients selected in each round of different T on two datasets with non-IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset



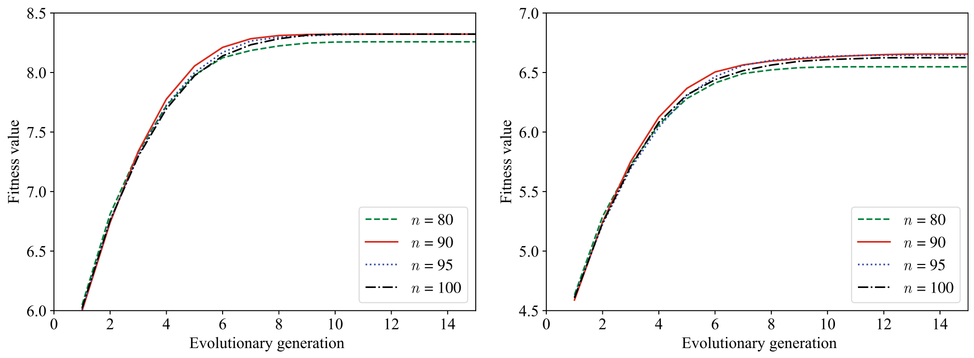
(a) Fashion-MNIST dataset (b) CIFAR-10 dataset

Fig. 13 Total number of selected clients of different T on two datasets with non-IID setting: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset



(a) Fashion-MNIST dataset (b) CIFAR-10 dataset

Fig. 14 Accuracy curves of different α on two datasets with non-IID setting and $T = 5$ min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset



(a) Fashion-MNIST dataset (b) CIFAR-10 dataset

Fig. 15 Fitness curves of different population size n with IID setting and T=3 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

Fig. 16 Fitness curves of different population size n with non-IID setting and T=5 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

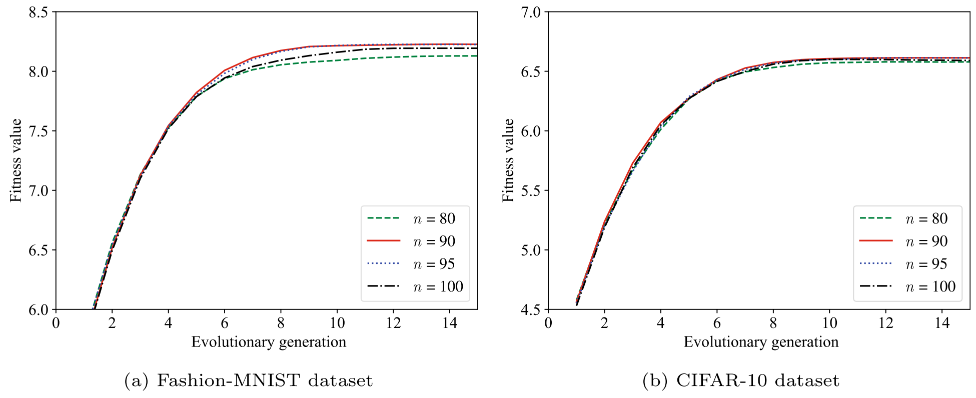


Fig. 17 Fitness heatmaps of k_1 and k_2 with IID setting and T=3 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

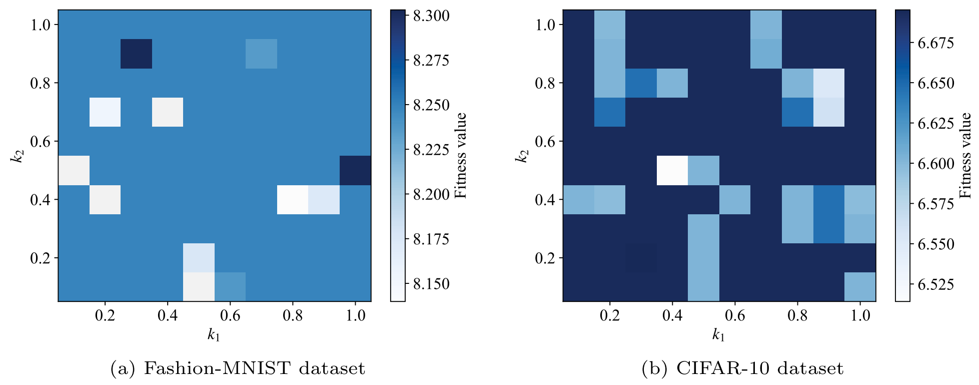
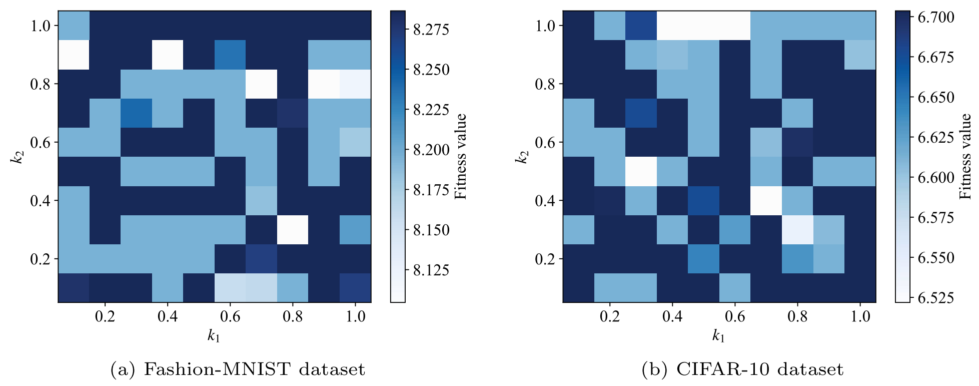


Fig. 18 Fitness heatmaps of k_1 and k_2 with non-IID setting and T=5 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset



5.2.4 Impacts of parameters of the genetic algorithm

In this section, we show the effect of parameters of the proposed genetic algorithm including the population size n, crossover probability constants k_1 and k_2, and mutation probability constants k_3 and k_4 under both IID and non-IID settings.

(1) Population size

Figures 15, 16 show the fitness curves of different population size n on two datasets with the IID and non-IID settings, respectively. It can be seen that the curves of fitness values over evolutionary generation are infected by different n. Too small n will lead to poor performance (i.e., low fitness value), while too large n will lead to long convergence time (i.e., large evolutionary generation). Therefore, according to the above results, the best performance of the proposed genetic algorithm is achieved when n takes 90.

(2) Crossover probability constants

Figures 17, 18 show the fitness heatmaps of k_1 and k_2 on two datasets with the IID and non-IID settings, respectively. It can be seen that all heatmaps are randomly distributed, and the difference between the maximum and minimum fitness values is less than 2.6%. This indicates that the strategy of adaptive crossover probability in this paper is not sensitive to k_1 and k_2 very much. When the

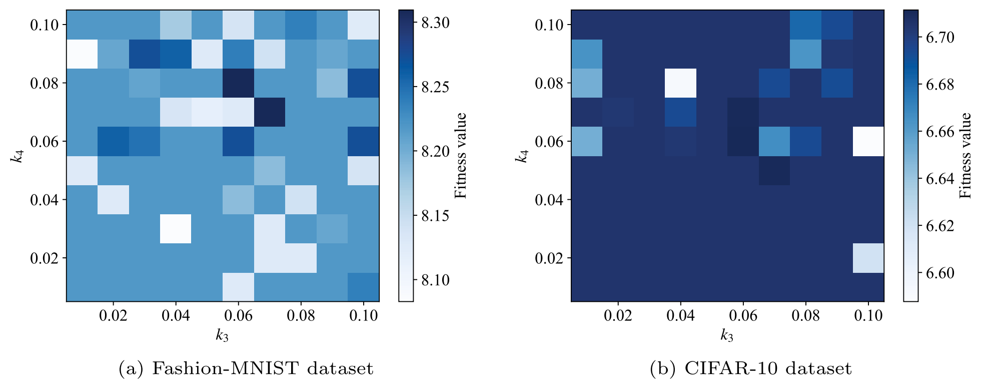


Fig. 19 Fitness heatmaps of k_3 and k_4 with IID setting and T=3 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

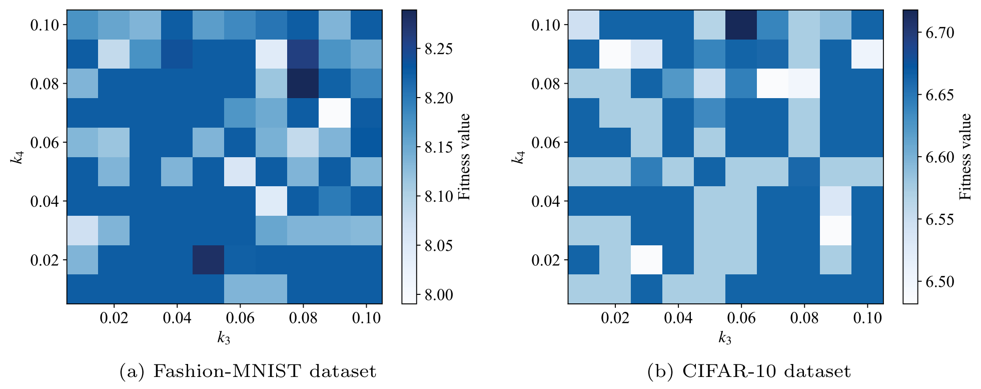


Fig. 20 Fitness heatmaps of k_3 and k_4 with non-IID setting and T=5 min: **a** Fashion-MNIST dataset **b** CIFAR-10 dataset

values of k_1 and k_2 are in a proper range, the proposed genetic algorithm can usually obtain stable solutions. Therefore, according to the typical crossover probability in genetic algorithm [34], we set $k_1 = 0.5$ and $k_2 = 0.9$ in this paper.

(3) Mutation probability constants

Figures 19, 20 show the fitness heatmaps of k_3 and k_4 on two datasets with the IID and non-IID settings, respectively. Similar to the crossover probability constants, the strategy of adaptive mutation probability is not sensitive to k_3 and k_4 as well. Therefore, we set a typical mutation probability in the genetic algorithm with $k_3 = 0.02$ and $k_4 = 0.05$ in this paper.

6 Conclusion

In this paper, we present FedCSGA, an optimized FL client selection algorithm based on the genetic algorithm to overcome the limitations of existing heuristic methods. By adaptive crossover and mutation operators and fine-tuning parameters, FedCSGA can increase the number of clients selected per round and improve the training performance of FL. Moreover, we integrate the model accuracy into the fitness function of FedCSGA to address non-IID data challenges while safeguarding client privacy. Experimental results demonstrate the superiority of our proposed FedCSGA over several state-of-the-art baseline FL algorithms in terms of the number of clients selected per round and model accuracy, where the first term is increased by 45.6% and 54.3%, the second term is increased by 2.4% and 7.7% on average in both IID and non-IID settings, respectively.

In the future, we will further study and solve the impacts of system and data heterogeneity among clients on FL training performance, and explore more effective modeling and scheduling methods on more real-world scenarios and datasets to achieve better model generalization and training effects. One interesting future work is to combine FL with reinforcement learning to adaptively explore the strategies and parameter settings of FedCSGA to improve the efficiency of client selection in FL. Another future work is to study personalized FL (PFL), whose goal is to learn individual models for each client rather than a single global model. The accuracy of individual models of clients will be more accurate in judging the performance of FL with data heterogeneity.

Author contributions Jiagao Wu: Conceptualization, Methodology, Supervision, Writing-review and editing. Hongyan Ji: Software, Validation, Writing-original draft. Jing Yi: Investigation, Data curation, Visualization. Linfeng Liu: Methodology, Supervision, Resources, Writing-review and editing.

Funding This research is supported by National Natural Science Foundation of China under Grants Nos. 62272237 and 61872191.

Data availability The datasets generated and/or analyzed during the current study are not publicly available due but are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

References

1. Li, P., Li, J., Huang, Z., Li, T., Gao, C., Yiu, S., Chen, K.: Multi-key privacy-preserving deep learning in cloud computing. Future Gener. Comput. Syst. **74**, 76–85 (2017)
2. Zeng, Z., Liu, Y., Tang, W.: Data-representation aware resource scheduling for edge intelligence. IEEE Trans. Vehicular Technol. **71**(12), 13372–13376 (2022)
3. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: The communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
4. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Int. Things J. **3**(5), 637–646 (2016)
5. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: 20th International Conference on Artificial Intelligence and Statistics, vol. 54, pp. 1273–1282 (2017). PMLR
6. Liu, Y., Kang, Y., Zou, T., Pu, Y., He, Y., Ye, X., Ouyang, Y., Zhang, Y.-Q., Yang, Q.: Vertical federated learning: concepts, advances, and challenges. IEEE Trans. Knowl. Data Eng. **36**(7), 3615–3634 (2024)
7. Zhu, B., Wang, L., Pang, Q., Wang, S., Jiao, J., Song, D., Jordan, M.I.: Byzantine-robust federated learning with optimal statistical rates. In: International Conference on Artificial Intelligence and Statistics, pp. 3151–3178 (2023). PMLR
8. Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., Avestimehr, A.S.: Federated learning for the internet of things: applications, challenges, and opportunities. IEEE Int. Things Magazine **5**(1), 24–29 (2022)
9. Yang, Z., Chen, M., Wong, K., Poor, H.V., Cui, S.: Federated learning for 6G: applications, challenges, and opportunities. Engineering **8**, 33–41 (2022)
10. Terrail, J., Leopold, A., Joly, C., Béguier, C., Andreux, M., Maussion, C., Schmauch, B., Tramel, E.W., Bendjebbar, E., Zaslavskiy, M., et al.: Federated learning for predicting histological response to neoadjuvant chemotherapy in triple-negative breast cancer. Nat. Med. **29**(1), 135–146 (2023)
11. Wang, T., Du, Y., Gong, Y., Choo, K.R., Guo, Y.: Applications of federated learning in mobile health: scoping review. J. Med. Internet Res. **25**, 43006 (2023)
12. Luo, B., Xiao, W., Wang, S., Huang, J., Tassiulas, L.: Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In: 2022 IEEE Conference on Computer Communications (INFOCOM), pp. 1739–1748 (2022). IEEE
13. Li, Q., Diao, Y., Chen, Q., He, B.: Federated learning on non-IID data silos: An experimental study. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 965–978 (2022). IEEE
14. Jamali-Rad, H., Abdizadeh, M., Singh, A.: Federated learning with taskonomy for non-IID data. IEEE Trans. Neural Netw. Learn. Syst. **34**(11), 8719–8730 (2023)
15. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: 2019 IEEE International Conference on Communications (ICC), pp. 1–7 (2019). IEEE
16. Yoshida, N., Nishio, T., Morikura, M., Yamamoto, K., Yonetani, R.: Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data. In: 2020 IEEE International Conference on Communications (ICC), pp. 1–7 (2020)
17. AbdulRahman, S., Tout, H., Mourad, A., Talhi, C.: FedMCCS: multicriteria client selection model for optimal IoT federated learning. IEEE Int. Things J. **8**(6), 4723–4735 (2020)
18. Xu, J., Wang, H.: Client selection and bandwidth allocation in wireless federated learning networks: a long-term perspective. IEEE Trans. Wireless Commun. **20**(2), 1188–1200 (2020)
19. Shi, W., Zhou, S., Niu, Z.: Device scheduling with fast convergence for wireless federated learning. In: 2020 IEEE International Conference on Communications (ICC), pp. 1–6 (2020). IEEE
20. Ching, C., Liu, Y., Yang, C., Kuo, J., Su, F.: Optimal device selection for federated learning over mobile edge networks. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pp. 1298–1303 (2020). IEEE
21. Wang, C., Yang, Y., Zhou, P.: Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity. IEEE Trans. Parallel Distrib. Syst. **32**(2), 394–410 (2020)
22. Zhang, W., Wang, X., Zhou, P., Wu, W., Zhang, X.: Client selection for federated learning with non-IID data in mobile edge computing. IEEE Access **9**, 24462–24474 (2021)
23. Lai, F., Zhu, X., Madhyastha, H.V., Chowdhury, M.: Oort: Efficient federated learning via guided participant selection. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 19–35 (2021). USENIX
24. Li, C., Zeng, X., Zhang, M., Cao, Z.: PyramidFL: A fine-grained client selection framework for efficient federated learning. In: 28th Annual International Conference on Mobile Computing and Networking (MOBICOM), pp. 158–171 (2022). ACM
25. Ribagin, S., Lyubenova, V.: In: Atanassov, K.T. (ed.) Metaheuristic algorithms: Theory and applications, pp. 385–419. Springer, Cham (2021)
26. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, USA (1992)
27. Kang, D., Ahn, C.W.: GA approach to optimize training client set in federated learning. IEEE Access **11**, 85489–85500 (2023)
28. Nguyen, T., Thai, M.T.: Preserving privacy and security in federated learning. IEEE ACM Trans. Netw. **32**(1), 833–843 (2024)
29. Shahid, O., Pouriyeh, S., Parizi, R.M., Sheng, Q.Z., Srivastava, G., Zhao, L.: Communication efficiency in federated learning: Achievements and challenges. Preprint at https://arxiv.org/abs/2107.10996 (2021)
30. Sesia, S., Toufik, I., Baker, M.: LTE-the UMTS long term evolution: from theory to practice. John Wiley & Sons, USA (2011)
31. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, 1979. W. H. Freeman & Co., USA (1979)
32. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 4th edn. MIT Press, USA (2022)
33. Kora, P., Yadlapalli, P.: Crossover operators in genetic algorithms: a review. Int. J. Comp. Appl. **162**(10), 34–36 (2017)
34. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994)
35. Soni, N., Kumar, T.: Study of various mutation operators in genetic algorithms. Int. J. Comp. Sci. Inform. Technol. **5**(3), 4519–4521 (2014)

36. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. Preprint at https://arxiv.org/abs/1806.00582 (2018)
37. Wang, H., Kaplan, Z., Niu, D., Li, B.: Optimizing federated learning on non-IID data with reinforcement learning. In: 2020 IEEE Conference on Computer Communications (INFOCOM), pp. 1698–1707 (2020). IEEE
38. Sattler, F., Müller, K.-R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2020)
39. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. Preprint at https://arxiv.org/abs/1708.07747 (2017)
40. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Handbook Syst. Autoimmune Dis. **1**(4), 1–58 (2009)

Jiagao Wu received the B.S. and M.S. degrees in physics from Nanjing University, Nanjing, China, in 1992 and 1995, respectively, and the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2006. Since May 2007, he has been an associate professor in the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include computer network, artificial intelligence, and mobile computing.

Hongyan Ji received the B.S. degree in computing science from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2021. At present, she is a master student of Nanjing University of Posts and Telecommunications, Nanjing, China. Her research interests include federated learning and mobile computing.

Jing Yi received the B.S. degree in information security from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2022. At present, she is a master student of Nanjing University of Posts and Telecommunications, Nanjing, China. Her research interests include federated learning and mobile computing.

Linfeng Liu received the B.S. and Ph.D. degrees in computer science from Southeast University, Nanjing, China, in 2003 and 2008, respectively. At present, he is a full professor of the School of Computer and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China. His main research interest lies in the areas of wireless networks and mobile computing. He has published more than 80 peer-reviewed papers in some technical journals or conference proceedings, such as IEEE Transactions on Parallel and Distributed Systems, ACM Transactions on Autonomous and Adaptive Systems, IEEE Transactions on Vehicular Technology, Computer Networks, Journal of Parallel and Distributed Computing, Journal of Network and Computer Applications, IEEE SECON, GLOBECOM, ICC, WCNC, CSCWD.