Genetic Algorithms for Federated Learning: Two Distinct Approaches

GenFed vs. FedCSGA - Comprehensive Analysis

Sarthak Mishra | Abhimanyu Singh Rathore

CS6007 Project

November 1, 2025

Overview

Federated Learning: The Core Problem

Given N clients with local datasets $\mathcal{D}_i$, the goal is to solve:

$$\min_{w \in \mathbb{R}^d} f(w) = \sum_{i=1}^{N} \frac{n_i}{n} F_i(w) \tag{1}$$

where $F_i(w) = \frac{1}{n_i} \sum_{(x,y) \in \mathcal{D}_i} \ell(w; x, y)$ is the local loss.

Key Challenges Addressed by these Papers:

- **Data Heterogeneity (Non-IID):** Clients' data distributions $\mathcal{D}_i$ vary significantly, causing local models to drift and slowing convergence.
- **System Heterogeneity:** Clients have different computation, memory, and network speeds.
- **Communication Bottlenecks:** Uploading models is slow; total training time is often constrained by a deadline.
- **Participant Reliability:** Some clients may be slow, drop out, or be malicious (Byzantine).

The Landscape of FL Client/Model Optimization

Our papers tackle the challenges of heterogeneity and efficiency. They fit into a broader landscape of FL optimization research:

Optimization-Based

- **Goal:** Maximize clients or minimize time.
- **FedCS:** Greedily selects clients who can meet a deadline.
- **Hybrid-FL:** Heuristic to select clients based on time and data distribution.
- **FedCSGA** Uses a full Genetic Algorithm to solve this NP-hard selection problem, outperforming simple greedy heuristics.

Utility-Function-Based

- **Goal:** Select clients with the highest "utility".
- **Oort:** Defines utility by training loss and time, prioritizing clients with high impact and speed.
- **PyramidFL:** A fine-grained utility approach.
- **GenFed** Uses a GA-inspired "fitness" (validation accuracy) to select *models* post-training, not *clients* pre-training.

Two GA Approaches: When to Apply?

FedCSGA (Wu et al.) *Genetic Client Selection*

- **When:** Before training round.
- **What:** Selects the optimal *set of clients* to participate.
- **Goal:** Maximize number of clients that can finish within a time deadline T.
- **GA Use:** Full GA (crossover, mutation) to solve the complex, NP-hard scheduling problem.

GenFed (Zheng et al.) *Genetic Model Aggregation*

- **When:** After clients train.
- **What:** Selects the optimal *set of models* to aggregate.
- **Goal:** Maximize global model quality by filtering out low-performing or malicious models.
- **GA Use:** GA-inspired (fitness, selection) but only uses the "Selection" step.

GenFed: Genetic Mechanism Mapping

Idea: FL as Genetic Evolution

Genetic Mechanism	Federated Learning Component
Population	Set of client models $\{w_i\}$
Crossover	Server aggregation $(\sum w_i)$
Mutation	Local client training (Gradient Descent)
Fitness Evaluation	**Model performance on a test set**
Selection	**Missing in traditional FL!**

GenFed's Contribution: Add the missing **Selection** step.

- **How:** Server maintains a small, public, and privacy-safe **global validation set** $\mathcal{D}_g^v$.
- Before aggregation, the server evaluates all k received models:

$$\text{fitness}_i = \text{Accuracy}(w_i^t, \mathcal{D}_g^v)$$

- It **selects** only the **top-**ρ_t models with the highest fitness.
- It **aggregates** only these elite models to form w_{t+1}.

GenFed: Pseudocode (Algorithm 2)

Server-Side Execution:
Initialize w_g^0, Global validation set $\mathcal{D}_g^v$
Initialization Phase: (Optional) Send supplemental data to clients with little data.
Training Phase:
for each round $t = 1, 2, \ldots, T$ **do**
    $S_t \leftarrow$ Select a subset of clients (e.g., 10 random clients)
    for each client $k \in S_t$ in parallel **do**
        Send w_g^t to client k
        $w_k^t \leftarrow$ ClientUpdate($w_g^t, \mathcal{D}_k$) {Local training = **Mutation**}
        Send w_k^t back to server
    end for
    — **GenFed Step** —
    for each received model w_k^t **do**
        fitness$_k \leftarrow$ Evaluate($w_k^t, \mathcal{D}_g^v$) {Evaluate fitness}
    end for
    $\rho_t \leftarrow$ GetAggregationQuantity($t, \rho_{\max}$, strategy)
    $C_t \leftarrow$ Select top-ρ_t models from S_t based on fitness {**Selection**}
    $w_g^{t+1} \leftarrow \sum_{i \in C_t} p_i w_i^t$ {**Crossover** (of elite models)}
end for
Return w_g^T

FedCSGA: System Model & Problem

Goal: Maximize clients selected, given a hard deadline T.

Time Model:

- Computation: $t_i^{\text{comp}} = (e \cdot d_i)/c_i$ (epochs $\times$ data / capacity)
- Communication: $t_i^{\text{comm}} = S/B_i$ (model size / bandwidth)
- Clients upload **sequentially** (one-by-one).

Key Insight: Computation can overlap with the *previous* client's communication. The total time depends on the **sequence q** of clients.

Optimization Problem:

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

- $\mathbf{q} = \langle q_1, q_2, \ldots \rangle$ is the client upload order.
- $\Theta^{\mathbf{q}}_{|\mathbf{q}|}$ is the total time for sequence **q** (a complex function of overlaps).
- This is a variant of Bin Packing, which is **NP-hard**.
- A simple greedy approach will fail; a GA is well-suited.

FedCSGA: Fitness Function

Chromosome: A sequence of clients, $\mathbf{q}_i = \langle q_{i1}, q_{i2}, \ldots \rangle$.
Fitness Function (with Adaptive Penalty):

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

- **Objective $h(\mathbf{q}_i)$:** How "good" is the sequence?
  - **IID:** $h(\mathbf{q}_i) = |\mathbf{q}_i|$ (Just maximize client count)
  - **Non-IID:** $h(\mathbf{q}_i) = \sum_{j=1}^{|\mathbf{q}_i|}(1 - \alpha \cdot A(q_{ij}))$ (Balances count with accuracy A, prioritizing low-accuracy clients)
- **Penalty $g(\mathbf{q}_i)$:** How "infeasible" is it?
  - $g(\mathbf{q}_i) = e^{\max(0,(\Theta - T)/T)} - 1$ (Exponential penalty for exceeding deadline T)
- **Penalty Factor λ:**
  - $\lambda = \lambda_0 e^{\sqrt{r}}$ (where r = generation)
  - **Intuition:** Start with low penalty (explore), end with high penalty (enforce deadline).

FedCSGA: Pseudocode (Algorithm 1, Part 1/2)

Input: Client set M, deadline T, population size n, generations R
Output: Optimal client sequence $\mathbf{q}^*$
Initialization:
$P_r \leftarrow \emptyset$ (population)
for $i = 1$ to n **do**
    $k \leftarrow$ random client from M
    $\mathbf{q}_i \leftarrow \langle k \rangle$
    Greedily add cheapest clients to $\mathbf{q}_i$ until $\Theta^{\mathbf{q}_i} > T$
    Add $\mathbf{q}_i$ to P_r
end for

FedCSGA: Pseudocode (Algorithm 1, Part 2/2)

Evolution:
while $r < R$ **do**
  $P_{r+1} \leftarrow \emptyset$
  Crossover:
  for $c = 1$ to $n/2$ **do**
    Select $\mathbf{q}_i, \mathbf{q}_j$ from P_r (e.g., tournament)
    $\mathbf{q}'_i, \mathbf{q}'_j \leftarrow \text{UniformCrossover}(\mathbf{q}_i, \mathbf{q}_j)$ with adaptive p_c
  end for
  Mutation:
  for $c = 1$ to n **do**
    $\mathbf{q}'_i \leftarrow \text{AdjacentSwapMutation}(\mathbf{q}_i)$ with adaptive p_m
  end for
  Selection:
  $P_{r+1} \leftarrow \text{SelectBest}(P_r \cup P', n)$ using fitness $F(\mathbf{q})$
  $r \leftarrow r + 1$
end while
$\mathbf{q}^* \leftarrow$ Best *feasible* chromosome ($\Theta \leq T$) from P_R
Return $\mathbf{q}^*$

Theoretical Analysis

GenFed (Zheng et al.):

- Contribution is conceptual (mapping FL to GA) , lacking a formal convergence proof.
- **Implied Guarantee:** Selecting models by validation accuracy biases the global model toward a better solution .
- This filtering also provides inherent robustness by discarding low-quality or Byzantine models

FedCSGA (Wu et al.):

- **NP-Hardness:** The paper formally states that the deadline-constrained, order-dependent client selection problem is **NP-hard**.
- This justifies using a meta-heuristic (like a GA) over a simple greedy algorithm (like FedCS), which is likely to get stuck in a local optimum.
- The GA is a tool to find a *high-quality approximate solution* to this NP-hard problem in polynomial time.

Base Training Configuration (Our Re-Implementation)

Model Architecture:

- MLP: 784 (input) $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 10 (output)
- Loss: CrossEntropyLoss
- Optimizer: SGD with momentum

Training Hyperparameters:

- Learning rate: $\eta = 0.01$
- Local epochs: $E = 5$
- Batch size: 32
- Device: CUDA/CPU auto-detect

Federated Setup:

- Datasets: MNIST
- Total Clients N: 10 (for our test) or 100 (in papers)
- Clients per Round k: 5 (our test) or 10 (in papers)
- Non-IID: Dirichlet distribution with α (lower α = more heterogeneous)

Key Hyperparameters Explained

GenFed (Zheng et al.):

- $\mathcal{D}_g^v$: **Global Validation Set**. A public, balanced dataset (e.g., 10% of MNIST test set) held by the server to evaluate model fitness.
- ρ_t: **Aggregation Quantity**. The number of "elite" models to select for aggregation.
- $\rho_{\max}$: **Max Aggregation**. The upper bound for ρ_t.
- **Strategy (Eq 5-9):** The function (Linear, Sinusoidal, etc.) that governs how ρ_t changes over time t.

FedCSGA (Wu et al.):

- T: **Time Deadline**. The hard constraint (in seconds) that the entire round (computation + sequential upload) must fit into.
- n: **Population Size**. The number of chromosomes (sequences) in the GA (e.g., 90).
- R: **Generations**. The number of evolution steps the GA takes.
- p_c, p_m: **Crossover/Mutation Probabilities**. Dynamically adapted based on chromosome fitness to balance exploration/exploitation.
- α: **Non-IID Factor**. A value in $[0, 1]$ that balances maximizing client

Paper Results: GenFed (Zheng et al.)

Setup: 100 clients, 10 selected/round, Dirichlet $\alpha = 0.1$

Performance (vs. FedAvg)

- **MNIST:** $15\times$ faster, $+1.17\%$ Acc
- **SVHN:** $34\times$ faster, $+1.62\%$ Acc
- **Fashion:** $36\times$ faster, $+7.71\%$ Acc
- **CIFAR-10:** $16\times$ faster, $+3.98\%$ Acc

Scalability

- FedAvg performance *degrades* as client count $\uparrow$
- GenFed performance *improves*

Byzantine Robustness (CIFAR-10)

Attack	FedAvg Drop	GenFed Drop
Label Flip	-5.90%	**-3.01%**
IPM	-2.93%	**-0.77%**
Mimic	-3.96%	**-1.38%**

Insight: The validation filtering naturally discards malicious models as "low fitness", reducing attack impact by $\geq 50\%$.

Paper Results: FedCSGA (Wu et al.)

Setup: 100 clients, GA (n=90, R=10), Non-IID ($\alpha = 0.7$)

IID Setting ($T = 3$min)

Method	Clients/Round	Accuracy
	Fashion-MNIST	
FedAvg	2.3	88.7%
FedCS	7.8	89.1%
FedCSGA	**8.5 (+9%)**	**89.9%**
	CIFAR-10	
FedAvg	1.6	70.6%
FedCS	6.5	72.3%
FedCSGA	**7.0 (+8%)**	**76.0%**

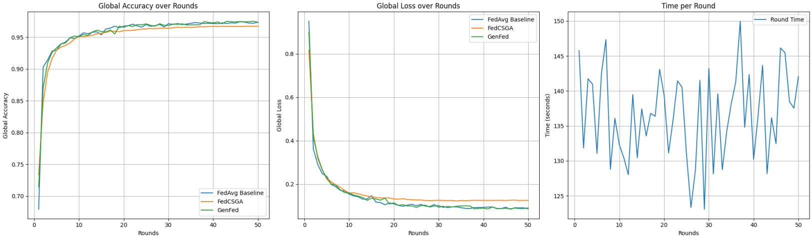
Non-IID Setting ($T = 5$min)

Method	Clients/Round	Accuracy
	Fashion-MNIST	
FedAvg	3.0	57.8%
PyramidFL	9.0	66.1%
FedCSGA	**13.5 (+50%)**	**66.5%**
	CIFAR-10	
FedAvg	2.9	38.8%
PyramidFL	7.5	53.3%
FedCSGA	**11.2 (+49%)**	**54.0%**

Key Insights:

- GA (FedCSGA) consistently finds better solutions (more clients) than the greedy (FedCS) or random (FedAvg) approaches.
- The Non-IID fitness function ($\alpha = 0.7$) is critical, dramatically improving accuracy over FedAvg (54.0% vs 38.8%) by balancing client count and data diversity.

Setup: $N = 10$ clients, $k = 5$ selected, MNIST, $\alpha = 0.5$



Method	Avg Accuracy
FedAvg (Baseline)	96.61%
GenFed ($\rho = 3$)	**97.02% (+0.41%)**
FedCSGA	96.18% (-0.43%)

Analysis:

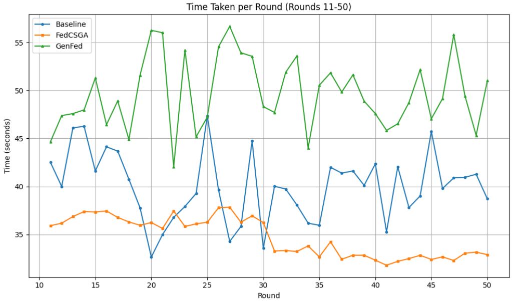
- **GenFed (Green):** Achieved the highest accuracy, confirming the paper's claim that filtering models improves quality.
- **FedCSGA (Orange):** Underperformed.
- **FedAvg (Blue):** Baseline.

Our Results: Client Selection Analysis

Analysis:

- **FedAvg (Left):** Selection is random, as expected. All 10 clients were selected roughly 20-30 times over 50 rounds.
- **FedCSGA (Center):** The GA was *highly* selective. It almost exclusively picked clients 0, 2, 4, 6, 8.
- **GenFed (Right):** Selection is random (like FedAvg) because GenFed selects *models* after training, it doesn't change the *client selection* policy.

Our Results: Timing Analysis (Extrapolated)



- **Time per Round (Left):** FedCSGA is stable; GenFed varies due to validation, FedAvg fluctuates moderately.
- **Time Ratios (Center):** GenFed slows with complexity; FedCSGA stays consistently fast.
- **Cumulative Time (Right):** FedCSGA is fastest overall; GenFed is slowest from validation overhead.

Lessons Learned: FedCSGA Failure Analysis

Root Cause: Stale Fitness Proxy

Problem in Our Implementation:

- FedCSGA's Non-IID fitness $h(\mathbf{q}) = \sum(1 - \alpha \cdot A(q_j))$ requires knowing client accuracy $A(q_j)$.
- We computed this accuracy **once** at round 0 (w_0).
- The GA used this *stale* accuracy as a proxy for all 50 rounds.
- As the global model w_t evolved, the round 0 accuracy became a meaningless proxy.
- The GA "overfit" to the stale proxy, picking clients (0,2,4,6,8) that were good at round 0, but not necessarily at round 40.

Proposed Fixes:

1. **Re-evaluate Fitness:** Re-run all clients on a validation set every $\sim$5 rounds to get fresh accuracy data for the GA.
2. **Scale Up:** The GA's advantage over greedy is minimal at $N = 10$. The papers used $N = 100$, where the search space is vast and a GA is necessary.

GenFed vs. FedCSGA: Key Differences

Aspect	GenFed	FedCSGA
Stage	**Post-training** (Model Aggregation)	**Pre-training** (Client Selection)
Decision	Which *models* to aggregate	Which *clients* to select
Objective	Maximize model quality/fitness	Maximize clients under a time deadline
GA Use	Minimal (Fitness + Selection only)	Full (Crossover, Mutation, Selection)
Chromosome	N/A (simple ranking)	A *sequence* of clients $\langle q_1, \ldots \rangle$
Fitness	Validation accuracy on $\mathcal{D}_g^v$	Function of $\lvert \mathbf{q} \rvert$, time Θ, and accuracy $A(\mathbf{q})$
Key Tool	Global validation set	Time model + Adaptive GA operators
Solves...	Data heterogeneity, Byzantine attacks	System heterogeneity, Time constraints

Verdict

Use GenFed when...

- ✓ You have a public, representative validation dataset.
- ✓ **Participant reliability** is a major concern (e.g., malicious clients, stragglers with buggy code).
- ✓ Your main goal is **fastest convergence** (fewer rounds) and highest final accuracy.
- × You have no hard per-round time deadlines.

Use FedCSGA when...

- ✓ You have **strict time deadlines** per round (e.g., real-time applications).
- ✓ **System heterogeneity** (client speeds) is the main bottleneck.
- ✓ You can accurately profile client computation (c_i) and bandwidth (B_i).
- × You do not have a public validation set to filter models.

Hybrid Approach: Use **FedCSGA** to select the max clients for a constraint T, then use **GenFed** to filter the received models.

Conclusion & Future Work

Summary

- **GenFed:** A simple, powerful *model selection* filter.
  - 10-35$\times$ faster convergence (in rounds)
  - Inherent Byzantine robustness
  - Validated in our re-implementation
- **FedCSGA:** A powerful *client selection* scheduler.
  - Solves NP-hard time-constrained selection
  - Needs scale ($N \gg 10$) and fresh fitness data

Future Work

- **Our Project:**
  - Fix FedCSGA fitness staleness
  - Scale to $N = 100$ clients
  - Implement and test the Hybrid approach
- **Research:**
  - Multi-objective GA for FedCSGA (time, accuracy, *and* fairness)

Code

https://github.com/sithtsar/FedGA

References

1. Zheng, H., Chu, J., Li, Z., Ji, J., & Li, T. (2025). "Accelerating Federated Learning with genetic algorithm enhancements." *Expert Systems With Applications, 281*, 127636.
2. Wu, J., Ji, H., Yi, J., & Liu, L. (2025). "Optimizing Client Selection in Federated Learning Base on Genetic Algorithm." *Cluster Computing, 28*, 400.
3. McMahan, B., et al. (2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data." AISTATS.
4. Nishio, T., & Yonetani, R. (2019). "Client selection for federated learning with heterogeneous resources in mobile edge." ICC.
5. Lai, F., et al. (2021). "Oort: Efficient federated learning via guided participant selection." OSDI.

Thank You Questions?

Backup: GenFed ρ_t Scheduling

Goal: Find the best strategy for varying ρ_t (number of models to aggregate).

- **Strategy (1):** Constant $\rho_t = \rho_{\max}$. (Stable, but slow start)
- **Strategy (3):** Linear $\rho_t = \min(\rho_{\max} \cdot t/c + 1, \rho_{\max})$
- **Strategy (4):** Sinusoidal $(\pi/2)$ $\rho_t = \min(\rho_{\max} \cdot \sin(t\pi/2c) + 1, \rho_{\max})$
- **Strategy (5):** Sinusoidal (π) (Unstable)

Result: Strategies (3) and (4) performed best.

- **Intuition:** Start with *small* ρ_t (e.g., $\rho_t = 1$) to aggregate only the *very* best model, allowing for rapid exploration.
- As training progresses, *increase* ρ_t to include more models, which provides stability and fine-tuning.

Backup: FedCSGA α Parameter Impact

Non-IID Fitness Function: $h(\mathbf{q}) = \sum_{j=1}^{|\mathbf{q}|}(1 - \alpha \cdot A(q_j))$

Table: Final Accuracy vs. α (Non-IID, $T = 5$min)

α	0.0	0.4	**0.7**	1.0
Fashion-MNIST	63.5%	65.2%	**66.5%**	66.1%
CIFAR-10	49.1%	52.2%	**54.0%**	53.3%

Analysis:

- $\alpha = 0$: Maximize clients only (same as IID). Ignores data heterogeneity.
- $\alpha = 1$: Only cares about accuracy. Selects too few clients, hurting diversification.
- $\alpha = 0.7$: **Optimal trade-off.** Balances selecting *many* clients (for diversity) with prioritizing clients that *need* training (low-accuracy clients).