异构边缘计算环境下异步联邦学习的节点分组与分时调度策略

马千飘[1]，贾庆民[1]，刘建春[2,3]，徐宏力[2,3]，谢人超[1,4]，黄韬[1,4]

（1. 网络通信与安全紫金山实验室未来网络研究中心，江苏 南京 211111；
2. 中国科学技术大学计算机科学与技术学院，安徽 合肥 230026；
3. 中国科学技术大学苏州高等研究院，江苏 苏州 215123；
4. 北京邮电大学网络与交换技术国家重点实验室，北京 100876）

摘 要：为了克服异构边缘计算环境下联邦学习的 3 个关键挑战，边缘异构性、非独立同分布数据及通信资源约束，提出了一种分组异步联邦学习（FedGA）机制，将边缘节点分为多个组，各个分组间通过异步方式与全局模型聚合进行全局更新，每个分组内部节点通过分时方式与参数服务器通信。理论分析建立了 FedGA 的收敛界与分组间数据分布之间的定量关系。针对分组内节点的通信提出了分时调度策略魔镜法（MMM）优化模型单轮更新的完成时间。基于 FedGA 的理论分析和 MMM，设计了一种有效的分组算法来最小化整体训练的完成时间。实验结果表明，FedGA 和 MMM 相对于现有最先进的方法能降低 30.1%～87.4%的模型训练时间。

关键词：边缘计算；联邦学习；非独立同分布数据；异构性；收敛分析

中图分类号：TP301

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023196

Client grouping and time-sharing scheduling for asynchronous federated learning in heterogeneous edge computing environment

MA Qianpiao[1], JIA Qingmin[1], LIU Jianchun[2,3], XU Hongli[2,3], XIE Renchao[1,4], HUANG Tao[1,4]

1. Future Network Research Center, Purple Mountain Laboratories, Nanjing 211111, China
2. School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China
3. Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou 215123, China
4. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: To overcome the three key challenges of federated learning in heterogeneous edge computing, i.e., edge heterogeneity, data Non-IID, and communication resource constraints, a grouping asynchronous federated learning (FedGA) mechanism was proposed. Edge nodes were divided into multiple groups, each of which performed global updated asynchronously with the global model, while edge nodes within a group communicate with the parameter server through time-sharing communication. Theoretical analysis established a quantitative relationship between the convergence bound of FedGA and the data distribution among the groups. A time-sharing scheduling magic mirror method (MMM) was proposed to optimize the completion time of a single round of model updating within a group. Based on both the theoretical analysis for FedGA and MMM, an effective grouping algorithm was designed for minimizing the overall training completion time. Experimental results demonstrate that the proposed FedGA and MMM can reduce model training time by 30.1%～87.4% compared to the existing state-of-the-art methods.

Keywords: edge computing, federated learning, Non-IID, heterogeneity, convergence analysis

收稿日期：2023−07−24；**修回日期：**2023−10−08

通信作者：刘建春，jcliu17@ustc.edu.cn

基金项目：国家自然科学基金资助项目（No.U1709217, No.61936015, No.92267301）

Foundation Item: The National Natural Science Foundation of China (No.U1709217, No.61936015, No.92267301)

0 引言

随着物联网的日益普及，物理世界每天都会产生大量的数据[1-2]。传统上，这些数据会被发送到远程云端进行训练或处理，长距离传输可能导致潜在的隐私泄露和大量的带宽消耗。因此，学界提出了边缘计算的范式，旨在利用网络边缘更广泛的计算能力，实现数据在网络边缘的高效处理。在边缘计算基础上，联邦学习（FL, federated learning）作为一种新的分布式机器学习范式，得到了应用和落地[3]。联邦学习采用的参数服务器（PS, parameter server）架构[4]由一个集中式参数服务器和大量的边缘节点组成。边缘节点在本地数据集上进行训练并将训练后的本地模型上传至参数服务器聚合，参数服务器将聚合后的全局模型下发返回边缘节点。由于边缘节点只上传其本地模型而不是原始数据，联邦学习能够有效保护边缘节点的隐私[5]。

然而，由于边缘环境的独特性，为了实现高效的联邦学习，需要考虑以下现实因素和挑战。1) 边缘异构性：各种异构的边缘设备都可参与模型训练，它们具有不同的数据量、地理位置、计算能力和信道状态[6]。2) 通信资源有限：边缘节点之间频繁的模型传输会消耗大量的通信带宽，由于通信资源有限，边缘网络很容易出现拥塞现象[7]。3) 非独立同分布（Non-IID, non-independent-and- identically-distributed）数据：由于边缘节点直接收集其所在区域内的数据，每个节点的本地数据之间通常是非独立同分布的[8]，极大影响联邦学习的性能。

已有的联邦学习更新机制分为同步和异步2种。在同步更新机制[3,6]中，参数服务器需要等待所有边缘节点的本地模型到达才进行聚合，因此单轮训练时间取决于最慢的节点。边缘异构性导致单轮更新时间长，这被称为straggler问题[9]，极大降低了模型整体训练速度。在异步更新机制[9-10]中，任一边缘节点的本地模型上传至参数服务器后，无须等待，直接与全局模型聚合。这导致边缘节点与参数服务器之间高频率的通信，造成了大量的通信资源消耗[11]；此外，由于数据在各个节点上的分布是Non-IID的，异步更新机制将导致模型向参与更新频率较高的节点偏移[12]，影响模型精度。

为克服已有同步和异步更新机制存在的缺点，本文提出了一种分组异步联邦学习（FedGA, grouping asynchronous federated learning）机制。如图1所示，FedGA将拥有不同数据量、数据分布的异构节点分为多组，并尽量使每个分组的数据分布接近全局数据分布，即接近独立同分布（IID）。若参数服务器收到来自同一组节点的本地模型，将这些本地模型与全局模型聚合为新的全局模型并将其下发至这一组的所有边缘节点。由于每个节点拥有不同的计算能力和通信状态，每个分组的单轮更新时间各不相同，因此各个分组异步地与参数服务器进行全局的模型更新。每个分组与参数服务器进行模型更新时，组内各个节点与参数服务器之间的通信采用分时通信方式[13-14]，通过设计的调度算法决策模型上传和下发的顺序，使尚未被调度的节点在其他节点传输模型时可以训练本地模型，实现了系统整体的“边算边传”，提升了对异构边缘的适应性。FedGA 能缓解同步机制中边缘异构带来的straggler 问题，且能减小异步机制中的资源消耗并应对Non-IID数据的不利影响，加快了模型训练速度并提升了模型训练精度。

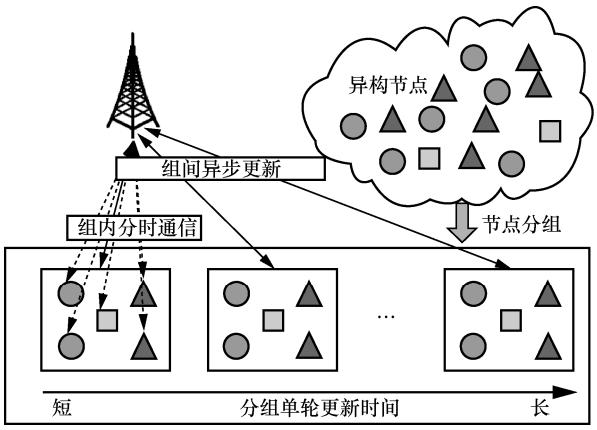

图1 分组异步联邦学习架构

本文的主要贡献如下。

1) 针对异构边缘场景，设计了FedGA机制，对其进行建模并分析了该机制的收敛性，分析了性能与数据分布之间的定量关系。

2) 为进一步应对边缘异构并提升通信效率，设计了一种新颖的分时调度策略魔镜法（MMM, magic mirror method）作为分组内部节点与参数服务器的通信策略，决策模型下发和上传顺序，最小化模型训练单轮更新时间。

3) 在对分组异步联邦学习的理论分析和分时调度策略的基础上，提出了一种节点分组算法，最小化模型训练整体完成时间。

4) 在经典数据集上的实验结果表明，在同精度

要求下，本文提出的 FedGA 和 MMM 相比已有研究能降低 30.1%～87.4%的模型训练时间。

1　系统模型和收敛分析

1.1　联邦学习的优化目标

典型的联邦学习架构由一个参数服务器和 N 个边缘节点 $\mathcal{V}=\{v_1,v_2,\cdots,v_N\}$ 组成。考虑一个 K 分类问题，类别空间为 $\mathcal{C}=\{c_1,c_2,\cdots,c_K\}$。设每个边缘节点 v_i 的本地数据集大小为 d_i，v_i 上类别为 c_k 的数据量为 d_i^k，则有

$$d_i=\sum_{c_k\in\mathcal{C}} d_i^k \tag{1}$$

所有边缘节点上的总数据量为

$$D=\sum_{v_i\in\mathcal{V}} d_i \tag{2}$$

设 $\alpha_i=\dfrac{d_i}{D}$ 为边缘节点 v_i 的数据占总数据量的比例，边缘节点标签为 c_k 的数据占总数据量的比例为

$$\gamma_k=\frac{\sum_{v_i\in\mathcal{V}} d_i^k}{D} \tag{3}$$

则广泛采用的交叉熵损失函数为

$$F(\boldsymbol{w})=\sum_{c_k\in\mathcal{C}} -\gamma_k\mathbb{E}_{\boldsymbol{x}|y=c_k}[\log p_k(\boldsymbol{x},\boldsymbol{w})] \tag{4}$$

其中，$\boldsymbol{w}$ 为模型参数，$(\boldsymbol{x},y)$为一个样本，$p_k(\boldsymbol{x},\boldsymbol{w})$ 为模型 $\boldsymbol{w}$ 预测输入样本 $\boldsymbol{x}$ 属于标签 c_k 的概率。类似地，边缘节点 v_i 在本地数据集的损失函数定义为

$$f_i(\boldsymbol{w})=\sum_{c_k\in\mathcal{C}} -\alpha_i^k\mathbb{E}_{\boldsymbol{x}|y=c_k}[\log p_k(\boldsymbol{x},\boldsymbol{w})] \tag{5}$$

其中，$\alpha_i^k=\dfrac{d_i^k}{d_i}$ 为边缘节点 v_i 标签为 c_k 的数据占其总数据量的比例。则本地和全局损失函数满足

$$F(\boldsymbol{w})=\sum_{v_i\in\mathcal{V}}\frac{d_i}{D}f_i(\boldsymbol{w})=\sum_{v_i\in\mathcal{V}}\alpha_i f_i(\boldsymbol{w}) \tag{6}$$

联邦学习优化目标为求解最优模型 $\boldsymbol{w}^*$ 以使全局损失函数 $F(\boldsymbol{w})$ 最小化，即 $\boldsymbol{w}^*=\operatorname{argmin}_{\mathrm{w}}F(\boldsymbol{w})$。

1.2　分组异步联邦学习

1.2.1　节点分组

将边缘节点集合 $\mathcal{V}$ 中的边缘节点分为 M 组 $\mathcal{V}_1,\cdots,\mathcal{V}_M$，每个边缘节点只能属于唯一的组，若节点 v_i 在分组 $\mathcal{V}_j$ 中，则 $x_{i,j}=1$；否则 $x_{i,j}=0$。设分组 $\mathcal{V}_j$ 中节点的数据量的总和为

$$D_j=\sum_{v_i\in\mathcal{V}_j} d_i=\sum_{v_i\in\mathcal{V}} x_{i,j}d_i \tag{7}$$

则分组 $\mathcal{V}_j$ 的数据占总数据量的比例为 $\beta_j=\dfrac{D_j}{D}$。设分组 $\mathcal{V}_j$ 中类别为 c_k 的数据量为

$$D_j^k=\sum_{v_i\in\mathcal{V}_j} d_i^k=\sum_{v_i\in\mathcal{V}} x_{i,j}d_i^k \tag{8}$$

则分组 $\mathcal{V}_j$ 中标签为 c_k 的数据占比为 $\beta_j^k=\dfrac{D_j^k}{D_j}$。节点分组后，每个分组与参数服务器之间进行异步聚合更新全局模型。

1.2.2　本地训练

当边缘节点 v_i 收到来自参数服务器的全局模型 $\boldsymbol{w}_{t-1}$ 时，其执行本地模型训练得到其本地模型

$$\boldsymbol{w}_t^i=\boldsymbol{w}_{t-1}-\eta\nabla f_i(\boldsymbol{w}_{t-1}) \tag{9}$$

其中，η 为学习率。接着，节点 v_i 将其本地模型 $\boldsymbol{w}_t^i$ 上传至参数服务器。

1.2.3　分组异步模型聚合

在参数服务器端，若第 t 轮接收到来自一个分组 $\mathcal{V}_j$ 的所有边缘节点本地模型，参数服务器执行第 t 轮模型聚合

$$\boldsymbol{w}_t=\left(1-\sum_{v_i\in\mathcal{V}_j}\alpha_i\right)\boldsymbol{w}_{t-1}+\sum_{v_i\in\mathcal{V}_j}\alpha_i\hat{\boldsymbol{w}}_t^i \tag{10}$$

其中，$\hat{\boldsymbol{w}}_t^i$ 为节点 v_i 参与第 t 轮模型聚合的本地模型。注意到，由于每个分组与参数服务器的模型聚合是异步的，$\hat{\boldsymbol{w}}_t^i$ 可能并不是本地模型 $\boldsymbol{w}_t^i$ [35]。设 τ_t 为当前轮次 t 与参与全局聚合的节点上一次接收到的全局模型版本之间的间隔，称为 staleness 因子[9]，则接收到的模型满足

$$\hat{\boldsymbol{w}}_t^i=\boldsymbol{w}_{t-\tau_t}^i \tag{11}$$

当参数服务器完成与分组 $\mathcal{V}_j$ 的模型聚合后，其把新的全局模型 $\boldsymbol{w}_t$ 分发给分组 $\mathcal{V}_j$ 中的所有节点。

图 2 给出了分组异步模型聚合的示例。6 个节点 v_1-v_6 被分为 3 组，边缘节点 v_1 和 v_2 在同一组，它们共同参与第一轮的模型聚合，即 $\boldsymbol{w}_1=(1-\alpha_1-\alpha_2)\boldsymbol{w}_0+\alpha_1\boldsymbol{w}_1^1+\alpha_2\boldsymbol{w}_1^2$，因此 $\hat{\boldsymbol{w}}_1^1=\boldsymbol{w}_1^1$ 且 $\hat{\boldsymbol{w}}_1^2=\boldsymbol{w}_1^2$，$\tau_1=0$；边缘节点 v_5 和 v_6 在同一组，它们共同参与第 3 轮的模型聚合，即 $\boldsymbol{w}_3=(1-\alpha_5-\alpha_6)\boldsymbol{w}_2+\alpha_5\boldsymbol{w}_1^5+\alpha_6\boldsymbol{w}_1^6$，因此 $\hat{\boldsymbol{w}}_3^5=\boldsymbol{w}_1^5$ 且 $\hat{\boldsymbol{w}}_3^6=\boldsymbol{w}_1^6$，$\tau_3=2$。

以上参数服务器分发全局模型到分组中节点和接收分组中节点的本地模型的过程都采用分时通信方式，并使用 MMM 分发和收集顺序，这将在第 2 节中详细阐述。

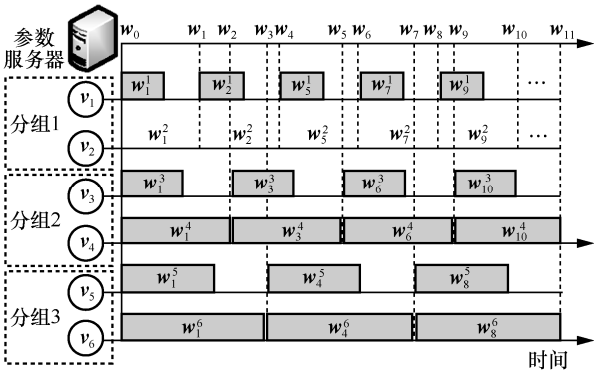

图 2　分组异步模型聚合的过程

1.3　收敛性分析

1.3.1　假设

本文对损失函数 $f_i, \forall v_i \in \mathcal{V}$ 做如下假设。

假设 1　平滑性。f_i 是 L-光滑的，即 $L>0$，对 $\forall \boldsymbol{w}_1, \boldsymbol{w}_2$，满足 $F_i(\boldsymbol{w}_2)-F_i(\boldsymbol{w}_1) \leqslant \langle \nabla F_i(\boldsymbol{w}_1), \boldsymbol{w}_2-\boldsymbol{w}_1\rangle+\frac{L}{2}\|\boldsymbol{w}_2-\boldsymbol{w}_1\|^2$。

假设 2　强凸性。f_i 是 μ-强凸的，即 $\mu \geqslant 0$，对 $\forall \boldsymbol{w}_1, \boldsymbol{w}_2$，满足 $F_i(\boldsymbol{w}_2)-F_i(\boldsymbol{w}_1) \geqslant \langle \nabla F_i(\boldsymbol{w}_1), \boldsymbol{w}_2-\boldsymbol{w}_1\rangle+\frac{\mu}{2}\|\boldsymbol{w}_2-\boldsymbol{w}_1\|^2$。

假设 3　梯度有界。每个节点数据的随机梯度的平方范数是一致有界的，即 $\forall k, \boldsymbol{w}$，有 $\mathbb{E}\|G_k(\boldsymbol{w})\|^2 \leqslant G^2$。其中 $G_k(\boldsymbol{w})=\nabla \mathbb{E}_{\boldsymbol{x}|y=c_k}[\log p_k(\boldsymbol{x}, \boldsymbol{w})]$。

假设 4　全局最优解。存在解 $\boldsymbol{w}^*$，使全局损失函数 $F(\boldsymbol{w})$ 达到最小，即 $\nabla F(\boldsymbol{w}^*)=0$。

1.3.2　收敛界分析

本文采用搬土距离（EMD, earth mover distance）来描述 2 个数据集 $\mathcal{D}_1$ 和 $\mathcal{D}_2$ 之间的数据分布的差异[8]。

$$\mathrm{EMD}(\mathcal{D}_1, \mathcal{D}_2)=\sum_{c_k \in \mathcal{C}}\left\|\frac{D_1^k}{D_1}-\frac{D_2^k}{D_2}\right\| \tag{12}$$

设 Γ_j 为分组 $\mathcal{V}_j$ 中边缘节点的总体数据与全局数据之间的 EMD，即

$$\Gamma_j=\mathrm{EMD}(\mathcal{D}, \mathcal{D}_j)=\sum_{c_k \in \mathcal{C}}\left\|\gamma_k-\beta_j^k\right\| \tag{13}$$

定理 1　若 $\boldsymbol{w}_0$ 为初始全局模型，$\eta<\frac{1}{L}$，则执行 t 轮全局聚合后的模型 $\boldsymbol{w}_T$ 满足

$$F(\boldsymbol{w}_T)-F(\boldsymbol{w}^*) \leqslant \rho^T(F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*))+\delta \tag{14}$$

其中，$\rho=\left(1-\mu\eta\sum_{j\in[1,M]}\psi_j\beta_j\right)^{\frac{1}{1+\tau_{\max}}}$，$\delta=\frac{\sum_{j\in[1,M]}\psi_j\beta_j\Gamma_j^2G^2}{2\mu\sum_{j\in[1,M]}\psi_j\beta_j}$。

定理 1 的详细证明参考文献[15]。

2　组内分时调度策略

2.1　分时策略描述

在联邦学习的聚合过程中，边缘节点通常采用分频（FS, frequency-sharing）通信方式（如 OFDMA[16]），或分时（TS, time-sharing）通信方式（如 TDMA[17]）与参数服务器通信。分频通信方式[18-19]将通信频率分配给多个边缘节点进行使用。然而，由于边缘节点具有异构性，而分配给边缘节点的频率是静态的，可能在某些时间段内被闲置，导致频谱资源的浪费[13]。分时通信方式[13-14]中，多个节点轮流使用同一个通信信道，在每个时间段内，只有一个边缘节点占用所有频谱资源上传或接收模型。分时通信方式相比分频具有更好的弹性，更能适应异构的边缘场景。然而，现有的基于分时通信方式的联邦学习研究只考虑了节点在模型上传到参数服务器时的调度问题，而假设模型下发的传输时延可忽略。此种这种假设仅在下行链路带宽远大于上行链路带宽的情况下才有效，例如，大功率基站作为参数服务器的场景下。而在小功率边缘节点作为参数服务器的联邦学习场景[20]下，模型下发可使用的带宽有限，造成的传输时延不可忽略，因此需要设计细粒度分时调度策略。本文综合考虑了异构边缘节点在模型下发、本地训练和模型上传时间方面的差异，提出了一种新的分时调度策略，决策模型下发和上传的顺序，以优化模型的单轮更新时间。

设某一个分组内有 N 个边缘节点，设 a_i 和 b_i 分别表示模型在边缘节点 v_i 和参数服务器之间下发时间和上传时间，c_i 表示模型在节点 v_i 上的训练时间。$\mathcal{G}=\left(O_{k_1}^{d_1}, \cdots, O_{k_n}^{d_n}, \cdots, O_{k_{2N}}^{d_{2N}}\right)$ 为一个调度策略，其中 $O_{k_n}^{d_n}$ 为策略的第 n 个操作，如果第 n 个操作为下发模型到边缘节点，则 $d_n=a$；如果第 n 个操作为上传模型至参数服务器，则 $d_n=b$。$k_1, k_2, \cdots, k_{2N}$ 为序列 $1,1,2,2,\cdots,N,N$ 的一个重排。只要一个调度策略 $\mathcal{G}$ 被确定，组内分时调度的完成时间就能相应地递推计算出来。设 T_n 为第 n 个操作的完成时间。

1) 如果 $O_{k_n}^{d_n}$ 为下发操作，即 $d_n=a$，则有 $T_n=T_{n-1}+a_{k_n}$。设 p_i 为边缘节点 v_i 完成本地训练的时间，则节点 v_{k_n} 完成本地训练的时间为下发时间加

训练时间，即

$$p_{k_n}=T_n+c_{k_n}=T_{n-1}+a_{k_n}+c_{k_n},\quad d_n=a \tag{15}$$

2) 如果 $O_{k_n}^{d_n}$ 为上传操作，即 $d_n=b$，其开始时间为 T_{n-1} 和 p_{k_n} 中的较大值，则其完成时间为 $T_n=\max\{T_{n-1},p_{k_n}\}+b_{k_n}$。

因此，策略每个操作完成时间的递推关系为

$$T_n=\begin{cases}0, & n=0\\ T_{n-1}+a_{k_n}, & 1\leqslant n\leqslant 2N, d_n=a\\ \max\{T_{n-1},p_{k_n}\}+b_{k_n}, & 1\leqslant n\leqslant 2N, d_n=b\end{cases} \tag{16}$$

策略 $\mathcal{G}$ 的完成时间为 T_{2N}，目标为获得最优的策略 $\mathcal{G}$ 以最小化其完成时间 T_{2N}。

图 3 给出了策略 $\mathcal{G}=\left(O_1^a,O_1^b,O_2^a,O_3^a,O_4^a,O_2^b,O_3^b,O_4^b\right)$ 在分组中有 4 个边缘节点的场景的执行过程。例如，$\mathcal{G}$ 的第一个操作是下发操作 O_1^a，即将模型下发到边缘节点 v_1，因此下发完成时间为 $T_1=T_0+a_1$，且模型在节点 v_1 完成本地训练的时间为 $p_1=T_0+a_1+c_1$。再如，$\mathcal{G}$ 的第 7 个操作是上传操作 O_3^b，即边缘节点 v_3 将其本地模型上传到参数服务器。由于第 6 个操作 O_2^b 的完成时间大于节点 v_3 完成本地模型训练的时间，即 $T_6>p_3$，第 7 个操作的完成时间为 $T_7=\max\{T_6,p_3\}+b_3=T_6+b_3$。$T_8$ 为调度策略 $\mathcal{G}$ 的完成时间。

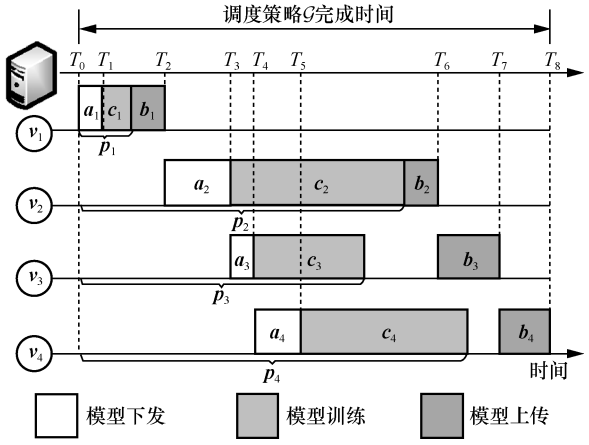

图 3　策略 $\mathcal{G}=\left(O_1^a,O_1^b,O_2^a,O_3^a,O_4^a,O_2^b,O_3^b,O_4^b\right)$ 的过程

2.2　魔镜法

本节提出分时调度策略魔镜法，其主要思想是首先随机获得一个下发序列和上传序列，然后交替在保持下发/上传序列不变的同时重排上传/下发序列，以最小化策略 $\mathcal{G}$ 的完成时间。

2.2.1　转化原调度策略

定理 2　将调度策略 $\mathcal{G}=\left(O_{k_1}^{d_1},O_{k_2}^{d_2},\cdots,O_{k_{2N}}^{d_{2N}}\right)$ 转化为一个前 N 个操作为下发操作、后 N 个操作为上传操作的新策略 $\hat{\mathcal{G}}=\left(O_{i_1}^a,O_{i_2}^a,\cdots,O_{i_N}^a,O_{j_1}^b,O_{j_2}^b,\cdots,O_{j_N}^b\right)$，其完成时间不超过原策略的完成时间。

定理 2 的证明如附录 1 所示。

图 4 给出了定理 2 的直观示例。对于调度策略 $\mathcal{G}=\left(O_1^a,O_1^b,O_2^a,O_3^a,O_4^a,O_2^b,O_3^b,O_4^b\right)$，其第 2 个操作 O_1^b 是上传操作，早于下发操作 O_2^a,O_3^a,O_4^a。通过定理 1，其被转化为新策略 $\hat{\mathcal{G}}=\left(O_1^a,O_2^a,O_3^a,O_4^a,O_1^b,O_2^b,O_3^b,O_4^b\right)$，如图 4(b)所示，新策略 $\hat{\mathcal{G}}$ 的完成时间显然小于 $\mathcal{G}$。

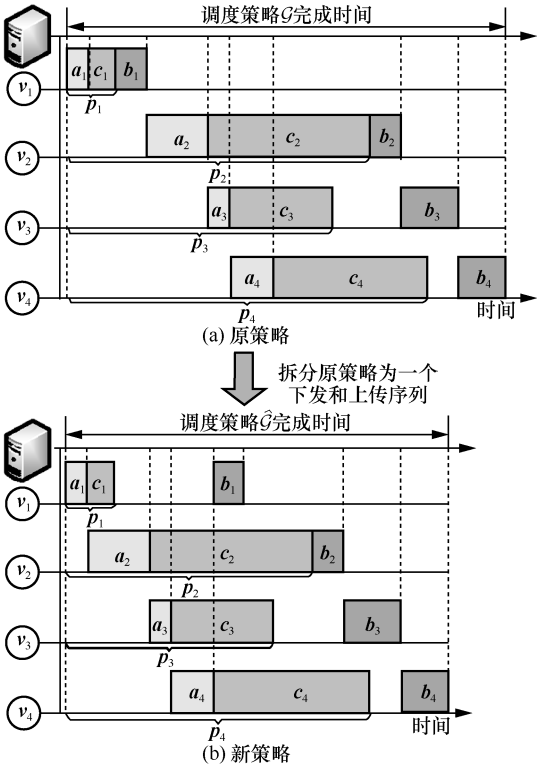

图 4　原策略转化为下发序列 O_1^a,O_2^a,O_3^a,O_4^a、上传序列 O_1^b,O_2^b,O_3^b,O_4^b 的新策略

2.2.2　重排上传序列

根据定理 2，构造一个前 N 个操作为下发操作 $O_{i_1}^a,O_{i_2}^a,\cdots,O_{i_N}^a$、后 N 个操作为上传操作 $O_{j_1}^b,O_{j_2}^b,\cdots,O_{j_N}^b$ 的策略 $\mathcal{G}$，其中 $i_1,i_2,\cdots,i_N$ 和 $j_1,j_2,\cdots,j_N$ 是序列 $1,2,\cdots,N$ 的重排。这样，可以重写式(15)和式(16)。设 $n\in[1,N]$，节点 v_{i_n} 上模型训练的完成时间为

$$p_{i_n}=\sum_{m=1}^{n}a_{i_m}+c_{i_n} \tag{17}$$

设 t_i 为节点 v_i 上传模型的完成时间，则节点 v_{j_n} 上传模型的完成时间为

$$t_{j_n}=\begin{cases}\sum_{i=1}^{N}a_i, & n=0\\ \max\{t_{j_{n-1}},p_{j_n}\}+b_{j_n}, & 1\leqslant n\leqslant N\end{cases} \tag{18}$$

由于操作 $O^b_{j_N}$ 是策略 $\mathcal{G}$ 的最后一个操作，$\mathcal{G}$ 的完成时间为 t_{j_N}。定理 3 给出了重排上传序列的方法。

定理 3　如果下发序列保持不变，将上传序列重排为 $O^b_{j_1},O^b_{j_2},\cdots,O^b_{j_N}$ 且满足 $p_{j_1}\leqslant p_{j_2}\leqslant\cdots\leqslant p_{j_N}$，将最小化策略 $\mathcal{G}$ 的完成时间。

证明　当下发序列保持不变时，调度问题等价于单机调度问题 $1|r|C_{\max}$ [21]，即最佳策略为按照每个节点的模型训练的完成时间 $p_{j_n},n\in[1,N]$ 从小到大的顺序进行调度。证毕。

图 5 给出了重排上传序列的直观示例。如图 5(a)所示，节点 v_1,v_2,v_3,v_4 上模型训练完成时间分别为 $p_1=a_1+c_1$，$p_2=a_1+a_2+c_2$，$p_3=a_1+a_2+a_3+c_3$，$p_4=a_1+a_2+a_3+a_4+c_4$，满足 $p_1\leqslant p_3\leqslant p_2\leqslant p_4$。因此，上传序列被重排为 O^b_1,O^b_3,O^b_2,O^b_4，如图 5(b)所示。

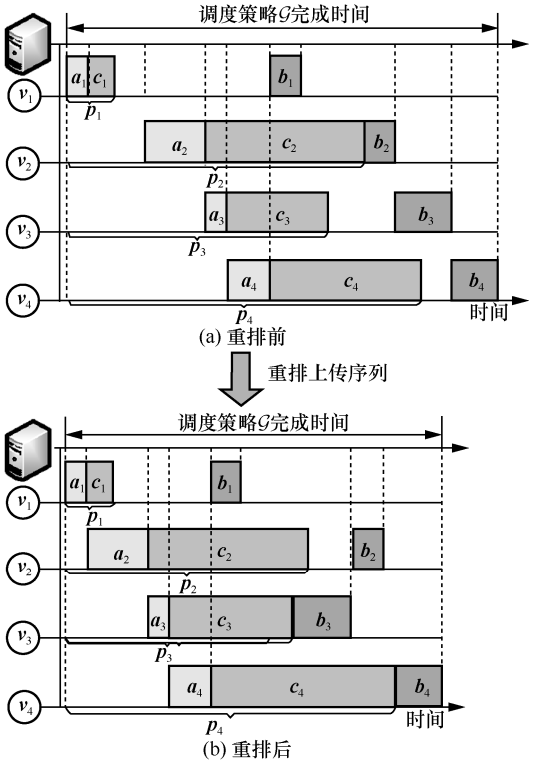

图 5　重排上传序列为 O^b_1,O^b_3,O^b_2,O^b_4

2.2.3　魔镜法的对称性

由定理 3 可知，当下发序列不变时，很容易确定最小化策略完成时间的最优上传序列；然而，当上传序列不变时，却很难确定最优的下发序列。在此引入镜像调度的概念来辅助解决此问题。

定义 1　镜像调度是原调度的对称形式，其下发序列是原调度上传序列的倒序，即 $O^b_{j_N},O^b_{j_{N-1}},\cdots,O^b_{j_1}$，其上传序列是原调度下发序列的倒序，即 $O^a_{i_n},O^a_{i_{N-1}},\cdots,O^a_{i_1}$。

例如，如图 6(a)所示，原调度的下发序列为 O^a_1,O^a_2,O^a_3,O^a_4，上传序列为 O^b_1,O^b_3,O^b_2,O^b_4。图 6(b)显示了相应的镜像调度，其下发序列为 O^b_4,O^b_2,O^b_3,O^b_1，上传序列为 O^a_4,O^a_3,O^a_2,O^a_1。从图 6 可以看出，一个调度策略与其镜像调度策略的完成时间相等。

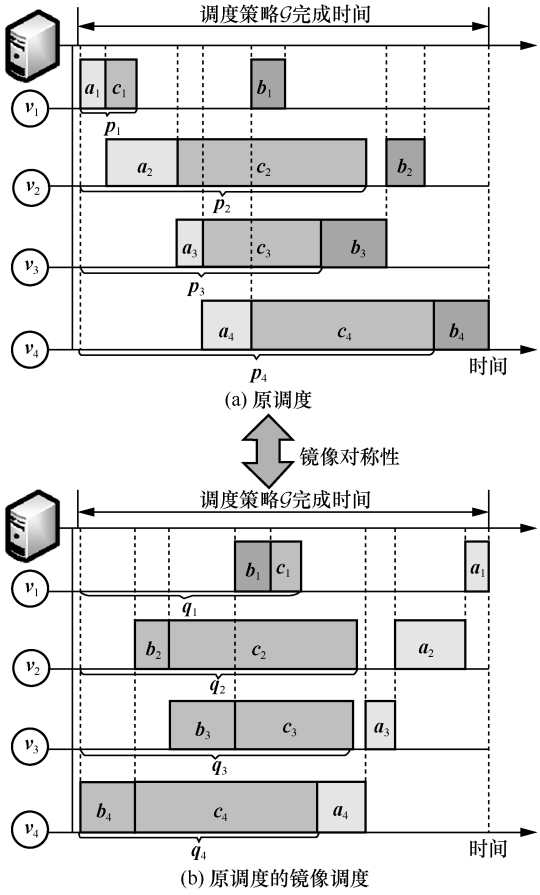

图 6　调度策略的镜像对称性

为建模镜像调度的完成时间，与式(17)和式(18)类似，可推导镜像调度完成时间的递推关系。依次设置 n 为 1 到 N，节点 v_{j_n} 上模型训练的完成时间为

$$q_{j_n}=\sum_{m=n}^{N}b_{j_m}+c_{j_n} \tag{19}$$

节点 v_{i_n} 上传模型的完成时间为

$$s_{i_n}=\begin{cases}\sum_{j=1}^{N} b_j, & n=N+1\\ \max\{s_{i_{n+1}}, q_{i_n}\}+a_{i_n}, & 1\leqslant n\leqslant N\end{cases} \tag{20}$$

由于操作 $O_{i_1}^a$ 是镜像策略的最后一个操作，其完成时间为 s_{i_1}。

定理 4　镜像对称性。一个调度策略 $\mathcal{G}$ 的完成时间与其镜像调度 $\hat{\mathcal{G}}$ 的完成时间相同。

定理 4 的详细证明如附录 2 所示。

2.2.4　重排下发序列

定理 5　如果上传序列保持不变，将下发序列重排为 $O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a$ 且满足 $q_{i_1}\geqslant q_{i_2}\geqslant\cdots\geqslant q_{i_N}$，将最小化策略 $\mathcal{G}$ 的完成时间。

证明　由定理 3 可知，如果调度 $\mathcal{G}$ 的镜像调度 $\hat{\mathcal{G}}$ 的下发序列 $O_{j_N}^b, O_{j_{N-1}}^b, \cdots, O_{j_1}^b$ 保持不变，重排镜像调度的上传序列为 $O_{i_N}^a, O_{i_{N-1}}^a, \cdots, O_{i_1}^a$ 且 $q_{i_N}\leqslant q_{i_{N-1}}\leqslant\cdots\leqslant q_{i_1}$ 会最小化镜像调度的完成时间。由定理 4 可知，调度 $\mathcal{G}$ 与其镜像调度 $\hat{\mathcal{G}}$ 的完成时间相等，因此重排 $\hat{\mathcal{G}}$ 的上传序列最小化 $\hat{\mathcal{G}}$ 的完成时间，相当于重排 $\mathcal{G}$ 的下发序列为 $O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a$ 且 $q_{i_1}\geqslant q_{i_2}\geqslant\cdots\geqslant q_{i_N}$，使 $\mathcal{G}$ 的完成时间最小化。证毕。

图 7 给出了重排下发序列的直观示例。图 7(b)是图 7(a)的一个镜像调度，节点 v_4, v_2, v_3, v_1 的模型训练完成时间分别为 $q_4=b_4+c_4$，$q_2=b_4+b_2+c_2$，$q_3=b_4+b_2+b_3+c_3$，$q_1=b_4+b_2+b_3+b_1+c_1$。满足 $q_1\leqslant q_4\leqslant q_3\leqslant q_2$，根据定理 3，镜像调度的上传序列（图 7(d)）被重排为 $O_1^a, O_4^a, O_3^a, O_2^a$ 以最小化镜像调度的完成时间，根据定理 4 可知，这相当于重排了原调度的下发序列为 $O_2^a, O_3^a, O_4^a, O_1^a$ 且最小化其完成时间。

2.2.5　魔镜法的整体流程

算法 1 显示了分组内节点调度策略的魔镜法的整体流程。首先随机生成一个下发序列 $O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a$ 和上传序列 $O_{j_1}^b, O_{j_2}^b, \cdots, O_{j_N}^b$。接着，进入一个多轮迭代过程，首先根据第 2.2.2 节中的方式重排上传序列（第 2～5 行）；然后根据第 2.2.4 节中的方式重排下发序列（第 6～9 行）；最后，计算当前调度序列的完成时间 $t_{j_N}^r$ 并判断算法是否收敛，如果 $t_{j_N}^r$ 等于上一轮调度的完成时间 t^*，则算法以收敛并返回最终调度策略，否则，置 $t^*=t_{i_N}^r$（第 10～14 行）。

算法 1　分组内节点调度策略魔镜法

输入　各节点模型下发、训练、上传时间 $a_i, b_i, c_i, \forall i$；随机初始化下发和上传序列

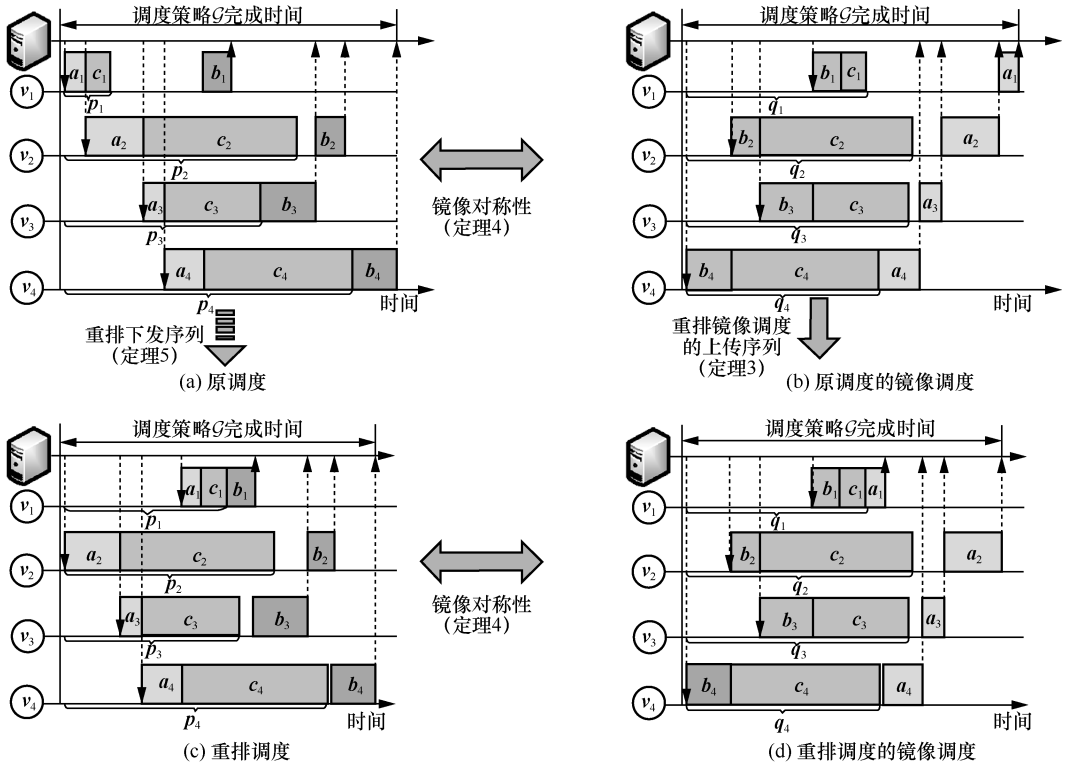

图 7　重排镜像调度的上传序列为 $O_1^a, O_4^a, O_3^a, O_2^a$ 相当于重排原序列下发序列为 $O_2^a, O_3^a, O_4^a, O_1^a$

$O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a$ 和 $O_{j_1}^b, O_{j_2}^b, \cdots, O_{j_N}^b$；初始化 $t^* = +\infty$

1) while True do
2) for $n = 1:1:N$
3) $p_{i_n} = \sum_{m=1}^{n} a_{i_m} + l_{i_n}$
4) 重排上传序列 $O_{j_1}^b, O_{j_2}^b, \cdots, O_{j_N}^b$ 使 $p_{j_1} \leqslant p_{j_2} \leqslant \cdots \leqslant p_{j_N}$
5) end for
6) for $n = N:-1:1$
7) $q_{j_n} = \sum_{m=n}^{N} b_{j_m} + l_{j_n}$
8) 重排下发序列 $O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a$ 使 $q_{i_1} \geqslant q_{i_2} \geqslant \cdots \geqslant q_{i_N}$
9) end for
10) $t_{j_0} = \sum_{i=1}^{N} a_i$
11) for $n = 1:1:N$
12) $t_{j_n} = \max\{t_{j_{n-1}}, p_{j_n}\} + b_{j_n}$
13) end for
14) 如果 $t_{j_N} < t^*$，$t^* = t_{j_N}$；否则返回调度策略 $\mathcal{G} = \left(O_{i_1}^a, O_{i_2}^a, \cdots, O_{i_N}^a, O_{j_1}^b, O_{j_2}^b, \cdots, O_{j_N}^b\right)$ 和完成时间 t^*

定理 6 MMM 能获得分时调度的一个局部最优策略，即无法通过单方面重排下发或上传序列使策略完成时间减小。

证明 由定理 2～定理 4 可知，对任意初始调度策略执行算法 1 后，不会增加策略完成时间，因此可通过迭代获得一个局部最优调度策略。证毕。

3 问题定义和节点分组算法

3.1 问题定义和转换

第 2 节介绍了组内分时调度策略，当所有节点的分组策略 $\boldsymbol{x} = \{x_{i,j} \mid 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant M\}$ 都确定后，则每个分组内的边缘节点都被确定，组内的分时调度策略完成时间可以由算法 2 获得，记分组 $\mathcal{V}_j$ 的组内完成时间为 $u_j = \mathrm{MMM}(a_i, b_i, c_i, \forall v_i \in \mathcal{V}_j)$。由于每个分组异步地参与全局模型的更新，平均一个轮次的完成时间为

$$\bar{u} = \frac{1}{\frac{1}{u_1} + \frac{1}{u_2} + \cdots + \frac{1}{u_M}} \tag{21}$$

因此，可定义优化问题为

$$\mathrm{P1}: \min \bar{u} T$$
$$\text{s.t.} \begin{cases} F(\boldsymbol{w}_T) \leqslant F(\boldsymbol{w}^*) + \epsilon \\ x_{i,j} \in \{0,1\}, \quad \forall i, j \end{cases}$$

其中，第一个不等式表示全局模型在第 T 轮后收敛到精度阈值 ϵ 以内。优化目标为确定节点分组，最小化全局模型训练完成时间。由定理 1 知 $F(\boldsymbol{w}_T) - F(\boldsymbol{w}^*) \leqslant \rho^T (F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)) + \delta$，其中 $\rho = \left(1 - \mu\eta \sum_{j \in [1,M]} \psi_j \beta_j\right)^{\frac{1}{1+\tau_{\max}}}$，$\delta = \frac{\sum_{j \in [1,M]} \psi_j \beta_j \Gamma_j^2 G^2}{2\mu \sum_{j \in [1,M]} \psi_j \beta_j}$。

令 $A \triangleq \frac{\epsilon - \delta}{F(\boldsymbol{w}_T) - F(\boldsymbol{w}^*)}$，则有

$$T \geqslant (1 + \tau_{\max}) \log_B A \tag{22}$$

其中，$B \triangleq 1 - \mu\eta \sum_{v_i \in \mathcal{V}} \psi_j \beta_j$。易知由于 staleness 因子最大的分组即组内完成时间 $u_j^{\max}$ 最长的分组，因此 $\tau_{\max}$ 可估计为

$$\hat{\tau}_{\max} = u_j^{\max} \left(\frac{1}{u_1} + \frac{1}{u_2} + \cdots + \frac{1}{u_M}\right) \tag{23}$$

根据以上分析，问题 P1 可转换为

$$\mathrm{P2}: \min \bar{u}(1 + \hat{\tau}_{\max}) \log_B A$$
$$\text{s.t.} \quad x_{i,j} \in \{0,1\}, \quad \forall i, j$$

3.2 节点分组策略算法

设 P2 的优化目标为 $U(\boldsymbol{x}) = \bar{u}(1 + \hat{\tau}_{\max}) \log_B A$。P2 有 2 个难点，首先，很难确认分组的个数 M；其次，即使分组数量 M 确定后，分组策略的搜索空间高达 $O(M^N)$，因此用穷举法找到使优化目标最小的分组策略是不现实的。本节设计一种贪心策略算法。如算法 2 所示，主要思路是将节点按数据量降序排列，再贪心地依次决策每个节点属于哪个分组，以使当前 $U(\boldsymbol{x})$ 的值最小。首先，初始化分组集合 $\mathcal{G}$ 为空集。接着，依次将 $\mathcal{V}$ 中的节点 v_i 分组到 $\mathcal{G}$ 中的分组或单独作为一组，比较不同分到不同组时 $U(\boldsymbol{x})$ 的值，然后将 v_i 分配至使 $U(\boldsymbol{x})$ 最小的分组（第 8 行）；如果 v_i 单独作为一个组使 $U(\boldsymbol{x})$ 最小，则为其单独创建一个分组（第 7 行）。当 $\mathcal{V}$ 中所有的节点被分组则算法停止。

算法 2 节点分组策略算法

定义分组集合 $\mathcal{G} = \varnothing$，初始化 $M = 1$，

$U_{\text{temp}}=+\infty$，$x_{i,j}=0,\forall i,j$。

1) for $v_i\in\mathcal{V}$
2) for $\mathcal{V}_j\in\mathcal{G}\cup\mathcal{V}_M$
3) $x_{i,j}=1$
4) 如果 $U(\boldsymbol{x})<U_{\text{temp}}$，则 $U_{\text{temp}}=U(\boldsymbol{x})$ 且 $j^*=j$
5) $x_{i,j}=0$
6) end for
7) 如果 $j^*==M$，则 $\mathcal{G}=\mathcal{G}\cup\mathcal{V}_M$ 且 $M=M+1$
8) $x_{i,j^*}=1$
9) 返回节点分组策略 $\boldsymbol{x}$
10) end for

4 实验

4.1 实验设置

4.1.1 环境设置

为验证提出的 FedGA 机制及其相关算法的性能，本节使用 Pytorch 框架下的深度学习库 PySyft 模拟了一个典型的边缘计算场景，包含一个参数服务器和多个边缘节点。边缘节点随机地部署在 50 m×50 m 的区域内，通过无线信道与位于一个区域中央的参数服务器通信。模型训练任务在一个 AMAX 深度学习工作站上执行。该 AMAX 工作站配备 8 核 Intel Xeon CPU(E52620v4)和 4 个显存为 11 GB 的 NVIDIA GeForce RTX 2080Ti GPU。实验环境为 Ubuntu 18.04，CUDA v10.0，cuDNN v7.5.0。

由于模型的训练实际都在同一个设备上完成，每个节点 v_i 完成本地训练的真实时间 $\hat{c}_i$ 类似，即 $\hat{c}_1=\hat{c}_2=\cdots=\hat{c}_N$。为了模拟边缘节点的异构性，设置每个节点的拥有不同的计算能力。具体地，设置节点 v_i 的模型训练时间为 $c_i=\kappa\hat{c}_i$，κ 为[1,5]之间的随机数，当 v_i 完成本地模型训练后，等待 0～4 倍训练时间再将模型发送给参数服务器。边缘节点 v_i 到参数服务器的模型上传速率由香农公式 $r_i^u=W_i^u\text{lb}\left(\frac{1+P_ih_i}{\sigma^2}\right)$ 给出，其中，W_i^u 为分配给节点 v_i 上传模型的带宽资源，所有上传模型的带宽资源为 $W^u=\sum_{v_i\in\mathcal{V}}W_i^u$，设置为 10 MHz；$P_i$ 为 v_i 的发送功率，设置为 100 mW；σ^2 为噪声功率，设置为−100 dBm。h_i 为节点到参数服务器的信道衰落，满足路径损耗模型 $h_i=h_0d_i^{-4}$ [22]，其中，$h_0=-40$ dBm 为路径损耗常数，d_i 为节点 v_i 到参数服务器的距离。设 ξ 为模型大小，则节点 v_i 到上传模型至参数服务器的时间为 $a_i=\frac{\xi}{r_i^u}$。设模型下发可用总带宽资源也为 10 MHz，同理可得模型下发时间 b_i。

4.1.2 模型和数据集

本文在 2 个经典数据集（MNIST[23]和 CIFAR -10[24]）上训练 2 个经典模型：逻辑回归模型（LR）和卷积神经网络（CNN）。为了分析联邦学习在不同数据分布下的训练性能，实验测试了 2 种划分数据集 MNIST 和 CIFAR-10 的方法：1) 数据独立同分布（IID），数据集中的所有数据混合后均匀地平分给所有边缘节点；2) 非独立同分布（Non-IID），为模拟数据在边缘节点的非独立同分布，本节采用标签倾斜法[20]将 MNIST 或 CIFAR-10 中的样本按标签分配给边缘节点。

4.1.3 基准机制和性能指标

为了验证 FedGA 在通信策略与节点选择策略上的优势，本文采用以下同步/异步/分组典型联邦学习机制作为基准与提出的 FedGA 机制对比。1) FedAvg[3]，一种同步联邦学习机制，每一轮所有节点参与模型聚合。2) FedAsy[9]，一种异步联邦学习机制，每个节点无须等待直接与全局模型聚合。3) TiFL[25]，节点按照与参数服务器的通信时间分为多组，分组间采用异步更新机制进行全局更新。

对于分组内聚合，选择 4 种算法与 MMM 做对比。1) 分频策略（FS），每个节点被分配静态带宽与参数服务器通信。2) 随机调度策略（Random），按随机顺序下发/上传模型。3) Up-Only[13-14]，在随机下发序列的基础上，决策上传序列最小化聚合完成时间。4) Optimal，将调度问题描述为一个非线性整数规划问题，使用求解器 CPLEX 获得最优解。

4.2 实验结果

4.2.1 分组内调度策略对比

本节模拟不同规模的边缘节点场景，测试了分组内调度策略的单轮聚合完成时间和调度算法运行时间。如图 8(a)所示，MMM 相对 FS 和 Random 能大幅减少分组内聚合完成时间且能获得接近 Optimal 的性能。例如，当分组内节点数为 100 时，Random、FS、MMM、Optimal 的分组内完成时间分别为 7.89 s、6.63 s、4.11 s、4.11 s，MMM 相比

Random 和 FS 减少了 47.9%和 38.0%的单轮聚合完成时间。当分组内节点数量大于 50 时，Up-Only 也能获得接近 Optimal 的性能，这是因为当节点数量较多时，模型训练的并行性较好，仅通过对上传序列的调度就获取一个最优的调度策略。而当分组内节点数量较少时，MMM 相比 Up-Only 有较大提升。例如，当分组内节点数量为 10 个时，Up-Only 和 MMM 的分组内聚合完成时间分别为 2.51 s 和 2.02 s，MMM 相比 Up-Only 降低聚合完成时间 19.5%。图 8(b)对比了 MMM 与 Optimal 的算法运行时间。MMM 的算法运行时间随着节点数量的增加线性增长，当分组内节点数量为 1 000 个时，运行时间仅为 6.35 ms，而 Optimal 的算法运行时间随着节点数量的增加呈指数级增长，当分组内节点数量为 100 个时，运行时间达到了 13 314.32 s。因此，在拥有大量边缘节点的场景下 MMM 在极短时间内就能获得接近最优的调度策略，而 Optimal 算法在拥有在大量边缘节点的场景无法实际部署。

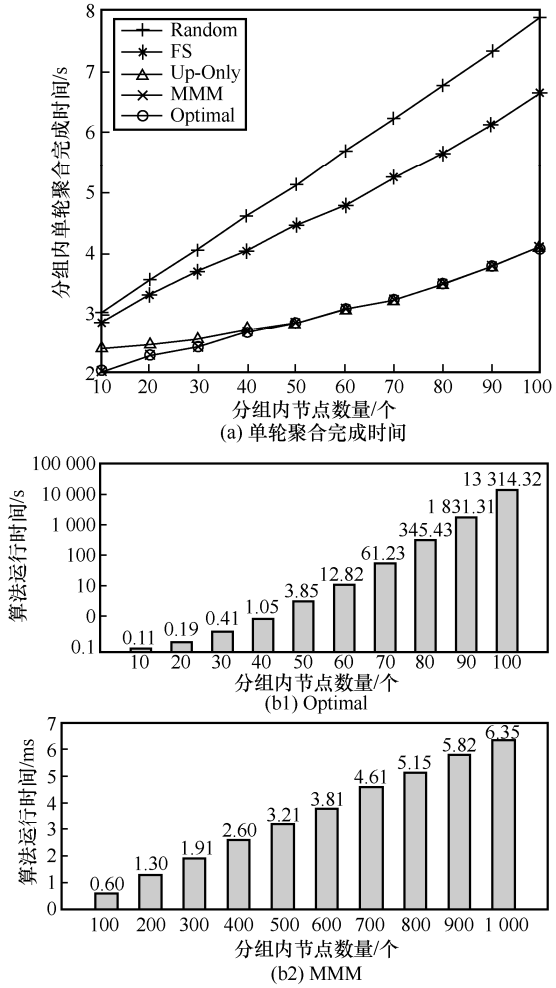

图 8　组内调度算法对比

4.2.2　与其他联邦学习机制对比

本节使用 PySyft 搭建了包含 100 个边缘节点的联邦学习的场景，测试 FedGA 与其他联邦学习机制 FedAvg、FedAsy 和 TiFL 的损失函数和精度。表 1 显示了通过 FedGA 和 TiFL 机制分组后各分组的平均 EMD。其中，$\bar{\Gamma}=\sum_{j\in[1,M]}\Gamma_j$ 表示不同分组机制对组间数据分布的影响。在 IID 设置下，不同类别的数据在每个节点上是平均分布的，节点在不同策略下分组后每个分组的数据分布都与数据的全局分布相同，因此在 FedGA 和 TiFL 下都有 $\bar{\Gamma}=0$。在 Non-IID 设置下，通过 FedGA 和 TiFL 机制分组后的平均 EMD 分别为 0.191 和 0.394。本文的 FedGA 在分组时尽量使分组内部的数据分布接近全局分布，而 TiFL 仅考虑了节点与参数服务器的通信时间，未考虑数据分布，因此 FedGA 相比 TiFL 更能使分组间的数据分布接近 IID。

表 1　通过 FedGA 和 TiFL 机制分组后各分组的平均 EMD

分布	FedGA	TiFL
IID	0	0
Non-IID	0.191	0.394

图 9～图 11 显示了在数据 IID 下不同联邦学习机制的损失函数和精度。从图 9～图 11 可知，在数据 IID 的情况下，不同机制的损失函数和精度曲线都比较平稳，而 FedGA 机制优于其他机制。例如，如图 10 所示，CNN 在数据集 MNIST 上训练 5 000 s 后，FedAvg、TiFL、FedAsy 和 FedGA 的精度分别为 91.7%、92.6%、93.2%、94.2%。FedAvg、TiFL、FedAsy 和 FedGA 达到 91%的精度所需要的时间分别为 3 882 s、2 959 s、1 770 s 和 1 362 s。与 FedAvg、TiFL 和 FedAsy 相比，FedGA 的训练时间分别减少了 64.9%、54.0%和 23.1%。FedAvg、TiFL、FedAsy 这 3 种机制效果提高的原因是，在数据 IID 下，每个节点上的数据分布相同，异步更新的方式造成的 staleness 问题不明显，而异步更新节点无等待时间，因此异步程度越高的机制模型训练效率越高。而本文的 FedGA 比 FedAsy 效率更高的原因是，节点采用了分时通信方式及 MMM，在异构边缘计算环境下能更高效地利用频谱资源。

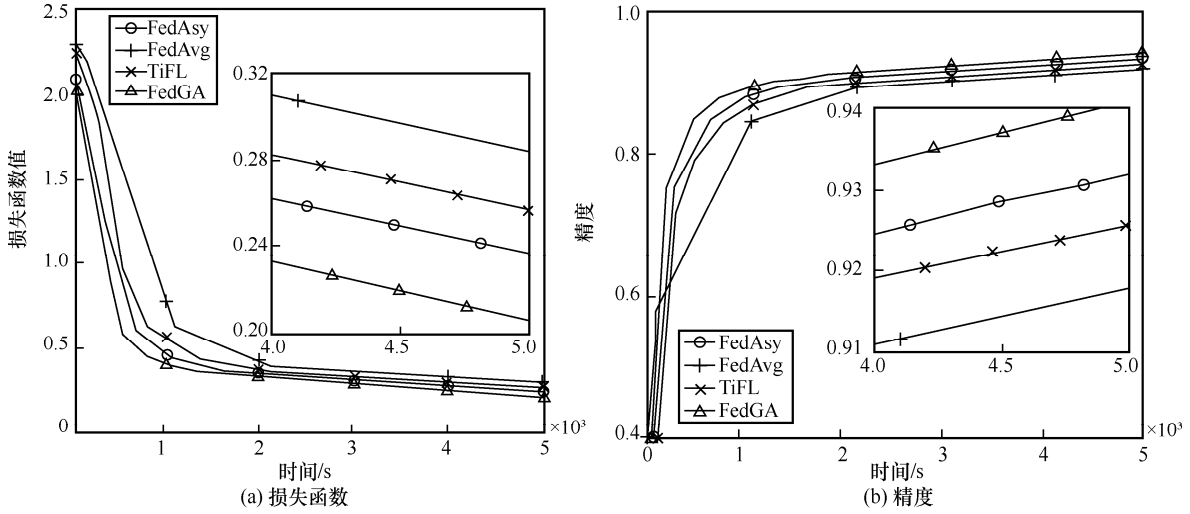

图 9　在数据 IID 下 LR 模型的损失函数和精度（MNIST）

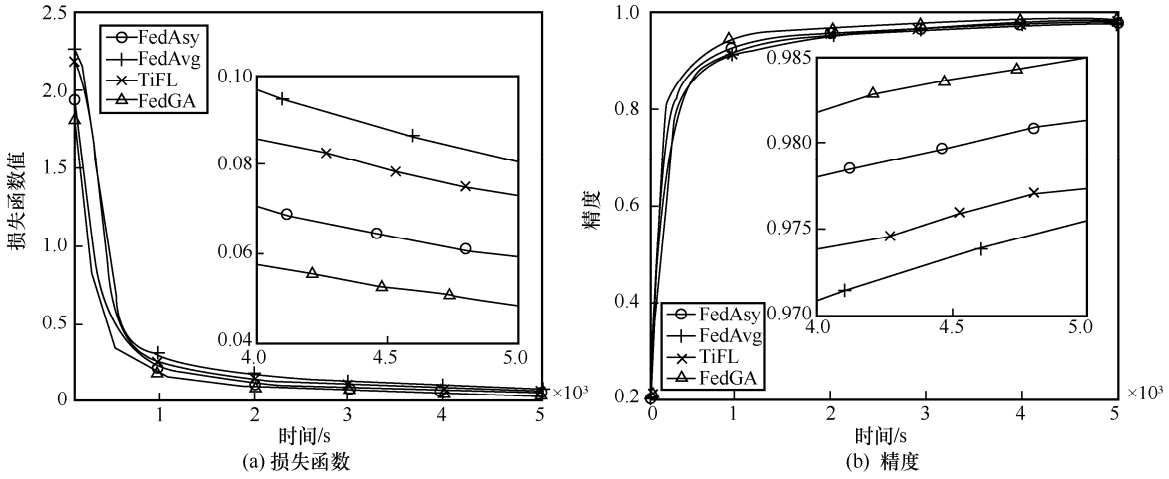

图 10　在数据 IID 下 CNN 模型的损失函数和精度（MNIST）

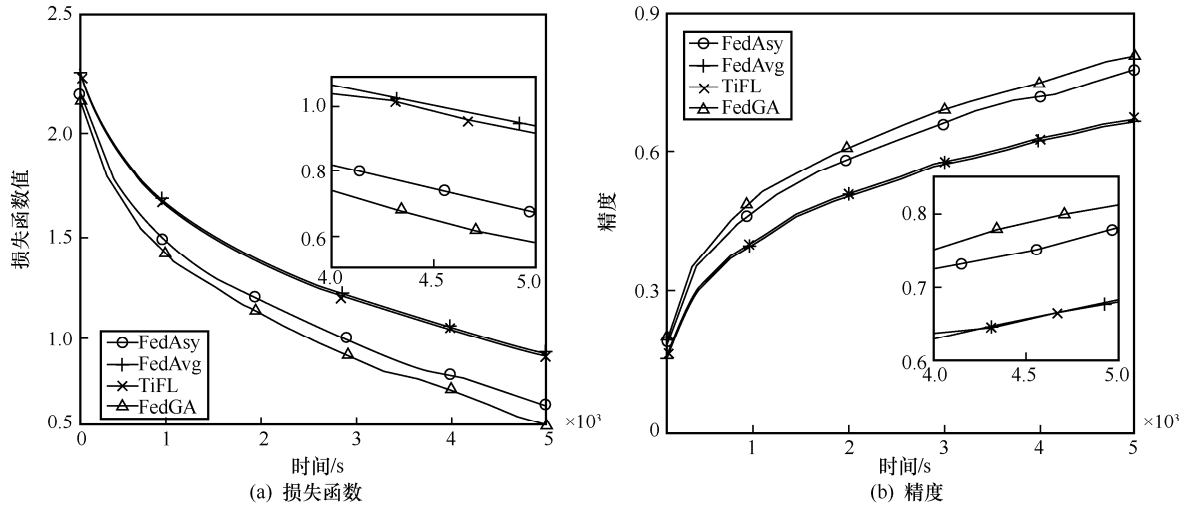

图 11　在数据 IID 下 CNN 模型的损失函数和精度（CIFAR-10）

图 12～图 14 显示了在数据 Non-IID 下不同联邦学习机制的损失函数和精度。从图 12～图 14 可知，FedAvg 和 FedGA 的损失函数和精度曲线比较平稳，抖动很小。FedAsy 的曲线抖动非常大，原因是其采用了异步机制更新全局模型，而边缘节点参与更新的频率差别很大，这导致部分

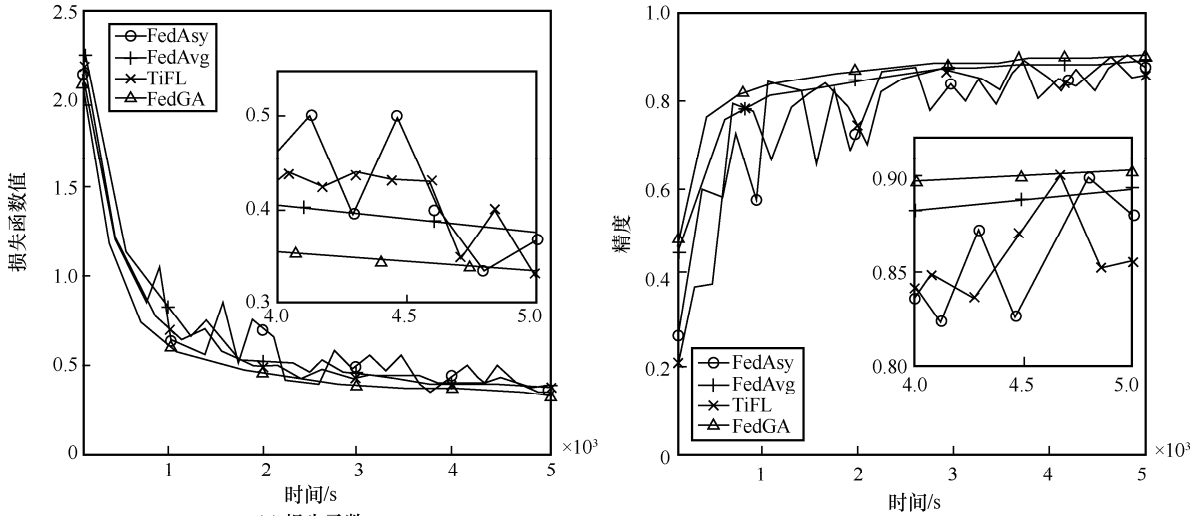

图 12　在数据 Non-IID 下 LR 模型的损失函数和精（MNIST）

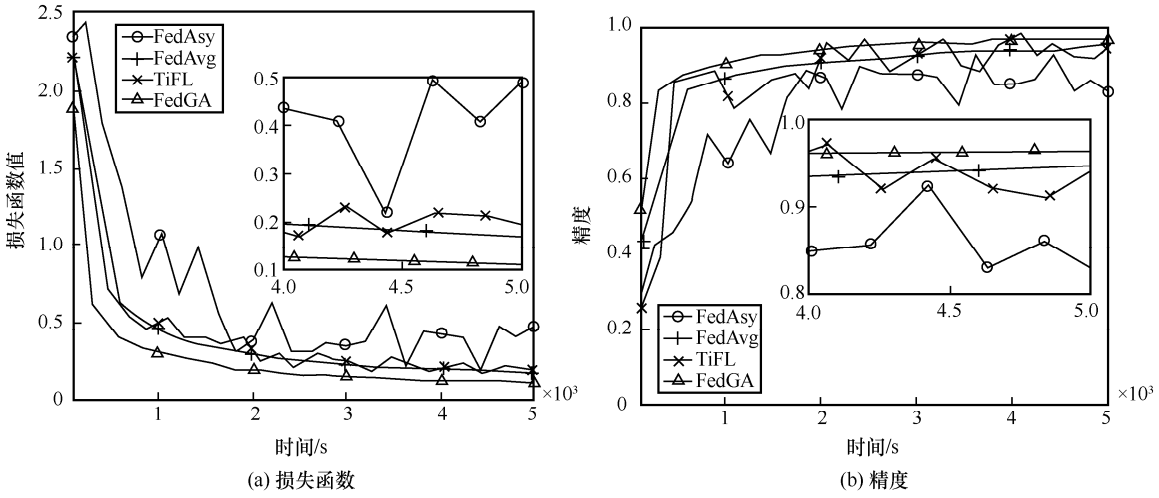

图 13　在数据 Non-IID 下 CNN 模型的损失函数和精度（MNIST）

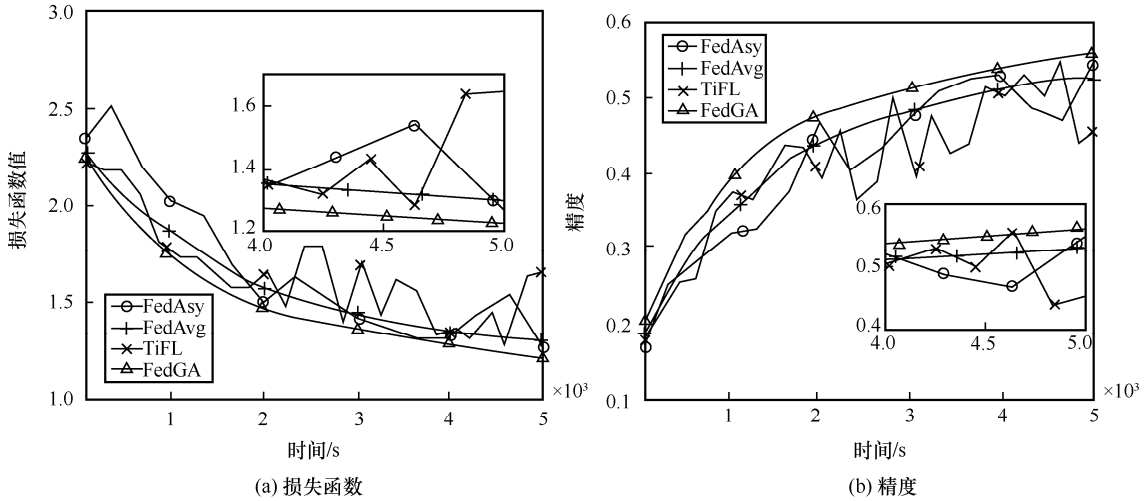

图 14　在数据 Non-IID 下 CNN 模型的损失函数和精度（CIFAR-10）

节点的 staleness 很大，而数据的 Non-IID 进一步加剧了陈旧模型带来的不利影响。FedAvg 曲线抖动小的原因是其采用了同步机制，因此本地模型能及时参与全局更新聚合为全局模型。本文的 FedGA 一方面将节点进行了分组，限制了节点本地模型的陈旧程度；另一方面考虑了节点数据分布，通过分组使分组之间的数据分布接近 IID，因此 FedGA 的更新过程也比较平稳。FedGA 相比其他联邦学习机制能在更短时间达到目标精度。例如，如图 12 所示，LR 在 MNIST 上训练时，FedAsy、TiFL、FedAvg 和 FedGA 稳定达到 80%精度需要的时间分别为 4 953 s、1 509 s、891 s 和 623 s。与 FedAsy、TiFL 和 FedAvg 相比，FedGA 的训练时间分别减少了 87.4%、58.7%和 30.1%。

上述实验结果表明，在不同数据分布下，本文的 FedGA 比其他机制有着更良好的训练性能，且能更好地应对边缘异构和通信资源受限的场景。

5　结束语

本文提出了适用于异构边缘环境下的 FedGA 机制，解决联邦学习面临的 3 个关键挑战，即边缘异构性、非独立同分布数据及通信资源约束。首先，对 FedGA 机制进行建模，并理论分析了数据分布对其收敛界的影响。接着，提出了分组内节点通信的分时调度策略魔镜法，进一步解决边缘异构和通信资源约束问题。最后，基于 FedGA 的收敛分析和 MMM，设计了一种有效的分组算法最小化模型整体训练时间。实验结果表明，FedGA 及 MMM 相对其他机制可以显著加快联邦学习的训练速度。

附录 1　定理 2 的证明

设策略 $\mathcal{G}$ 的第 n 个操作为上传操作，第 $n+1$ 个操作为下发操作。根据式 (16)，$O_{k_n}^b$ 的完成时间为 $T_n=\max\{T_{n-1},p_{k_n}\}+b_{k_n}$，则 $O_{k_{n+1}}^a$ 的完成时间为

$$\begin{aligned}&T_{n+1}=T_n+a_{k_{n+1}}=\max\{T_{n-1},p_{k_n}\}+b_{k_n}+a_{k_{n+1}}=\\&\max\{T_{n-1}+a_{k_{n+1}},p_{k_n}+a_{k_{n+1}}\}+b_{k_n}\end{aligned}\tag{24}$$

交换策略 $\mathcal{G}$ 中操作 $O_{k_n}^b$ 和 $O_{k_{n+1}}^a$ 的次序获得策略 $\hat{\mathcal{G}}$，则 $\hat{\mathcal{G}}$ 中第 n 个操作 $O_{k_{n+1}}^a$ 的完成时间为 $\hat{T}_n=T_{n-1}+a_{k_{n+1}}$，$\hat{\mathcal{G}}$ 中第 $n+1$ 个操作的完成时间变为

$$\hat{T}_{n+1}=\max\{\hat{T}_n,p_{k_n}\}+b_{k_n}=\max\{T_{n-1}+a_{k_{n+1}},p_{k_n}\}+b_{k_n}\tag{25}$$

显然有 $T_{n+1}\geqslant\hat{T}_{n+1}$。类似地，可推导得到 $T_{n+2}\geqslant\hat{T}_{n+2}$，$T_{n+3}\geqslant\hat{T}_{n+3},\cdots,T_{2N}\geqslant\hat{T}_{2N}$。因此，策略 $\hat{\mathcal{G}}$ 的完成时间不大于策略 $\mathcal{G}$ 的完成时间。证毕。

附录 2　定理 4 的证明

考虑调度 $\mathcal{G}=\left(O_{i_1}^a,O_{i_2}^a,\cdots,O_{i_N}^a,O_{j_1}^b,O_{j_2}^b,\cdots,O_{j_N}^b\right)$，其下发序列为 $O_{i_1}^a,O_{i_2}^a,\cdots,O_{i_N}^a$，上传序列为 $O_{j_1}^b,O_{j_2}^b,\cdots,O_{j_N}^b$。设 $\mathcal{P}(n)$ 为 $O_{j_n}^a$ 在其下发序列中的位置，$\mathcal{P}'(n)$ 为 $O_{i_n}^b$ 在其上传序列中的位置。显然，对于 $1\leqslant n\leqslant N$，有 $\mathcal{P}(\mathcal{P}'(n))=n$ 和 $\mathcal{P}'(\mathcal{P}(n))=n$。由式(18)可得 $\mathcal{G}$ 的完成时间为

$$\begin{aligned}&T_{2N}=t_{j_N}=\max\{t_{j_{N-1}},p_{j_N}\}+b_{j_N}=\\&\max\{t_{j_{N-1}}+b_{j_N},p_{j_N}+b_{j_N}\}=\\&\max\{\max\{t_{j_{N-2}},p_{j_{N-1}}\}+b_{j_{N-1}}+b_{j_N},\\&p_{j_N}+b_{j_N}\}=\max\{t_{j_{N-2}}+b_{j_{N-1}}+b_{j_N},\\&p_{j_{N-1}}+b_{j_{N-1}}+b_{j_N},p_{j_N}+b_{j_N}\}=\cdots=\\&\max\left\{t_{j_0}+\sum_{n=1}^{N}b_{j_n},p_{j_1}+\right.\\&\sum_{n=1}^{N}b_{j_n},p_{j_2}+\sum_{n=2}^{N}b_{j_n},\cdots,p_{j_{N-1}}+\\&\left.b_{j_{N-1}}+b_{j_N},p_{j_N}+b_{j_N}\right\}\end{aligned}\tag{26}$$

由式(17)可知，对于 $n\in[1,N]$，有 $p_{i_n}=\sum_{m=1}^{n}a_{i_m}+c_{i_n}$。此外，显然有 $t_{j_0}=\sum_{i=1}^{n}a_i$ 和 $\sum_{n=1}^{n}b_{j_n}=\sum_{i=1}^{n}b_i$，因此

$$\begin{aligned}&T_{2N}=\max\left\{\sum_{i=1}^{n}(a_i+b_i),p_{j_1}+\sum_{n=1}^{n}b_{j_n},p_{j_2}+\right.\\&\left.\sum_{n=2}^{n}b_{j_n},\cdots,p_{j_N}+b_{j_N}\right\}=\max\left\{\sum_{i=1}^{n}(a_i+b_i),\right.\\&\left.\sum_{m=1}^{\mathcal{P}(1)}a_{i_m}+c_{j_1}+\sum_{m=1}^{N}b_{j_m},\sum_{m=1}^{\mathcal{P}(2)}a_{i_m}+c_{j_2}+\sum_{m=2}^{N}b_{j_m},\cdots,\right.\\&\left.\sum_{m=1}^{\mathcal{P}(N)}a_{i_m}+c_{j_N}+b_{j_N}\right\}=\\&\max\left\{\sum_{i=1}^{n}(a_i+b_i)\right\}\cup Q\end{aligned}\tag{27}$$

其中，$Q=\left\{\sum_{m=1}^{\mathcal{P}(n)}a_{i_m}+c_{j_n}+\sum_{m=n}^{N}b_{j_n}\,|\,1\leqslant n\leqslant N\right\}$，$T_{2N}$ 是集合 $\left\{\sum_{i=1}^{N}(a_i+b_i)\right\}\cup Q$ 中的最大值。由式(20)可得镜像调度 $\hat{\mathcal{G}}$ 的完成时间为

$$\hat{T}_{2N}=s_{i_1}=\max\{s_{i_2},q_{i_1}\}+a_{i_1}=$$
$$\max\{s_{i_2}+a_{i_1},q_{i_1}+a_{i_1}\}=$$
$$\max\{\max\{s_{i_3},q_{i_2}\}+a_{i_2}+a_{i_1},q_{i_1}+a_{i_1}\}=$$
$$\max\{s_{i_3}+a_{i_2}+a_{i_1},q_{i_2}+a_{i_2}+a_{i_1},q_{i_1}+a_{i_1}\}=\cdots=$$
$$\max\left\{s_{i_{N+1}}+\sum_{n=1}^{N}a_{i_n},q_{i_N}+\sum_{n=1}^{N}a_{i_n},q_{i_{N-1}}+\sum_{n=1}^{N-1}a_{i_n},\cdots,q_{i_1}+a_{i_1}\right\} \quad (28)$$

由式(19)可知，对于 $n\in[1,N]$，有 $q_{j_n}=\sum_{m=n}^{N}b_{j_m}+c_{j_n}$。此外，显然有 $s_{i_{N+1}}=\sum_{i=1}^{N}b_i$ 和 $\sum_{n=1}^{N}a_{i_n}=\sum_{i=1}^{N}a_i$，因此

$$\hat{T}_{2N}=\max\left\{\sum_{i=1}^{N}(b_i+a_i),q_{i_N}+\sum_{n=1}^{N}a_{i_n},q_{i_{N-1}}+\sum_{n=1}^{N-1}a_{i_n},\cdots,q_{i_1}+a_{i_1}\right\}=$$
$$\max\left\{\sum_{i=1}^{N}(b_i+a_i),\sum_{m=\mathcal{P}'(N)}^{N}b_{j_m}+c_{i_N}+\sum_{m=1}^{N}a_{i_m},\sum_{m=\mathcal{P}'(N-1)}^{N}b_{j_m}+c_{i_{N-1}}+\sum_{m=1}^{N-1}a_{i_m},\cdots,\sum_{m=\mathcal{P}'(1)}^{N}b_{j_m}+c_{i_1}+a_{i_1}\right\}=\max\left\{\sum_{i=1}^{N}(b_i+a_i)\right\}\cup\hat{Q} \quad (29)$$

其中，$\hat{Q}=\left\{\sum_{m=\mathcal{P}'(n)}^{N}b_{j_m}+c_{i_n}+\sum_{m=1}^{n}a_{i_m}\,|\,1\leqslant n\leqslant N\right\}$，$\hat{T}_{2N}$ 是集合 $\left\{\sum_{i=1}^{N}(b_i+a_i)\right\}\cup\hat{Q}$ 中的最大值。考虑集合 $\hat{Q}$ 中的任一个元素 $\hat{Q}_n$，其满足

$$\hat{Q}_n=\sum_{m=\mathcal{P}'(n)}^{N}b_{j_m}+c_{i_n}+\sum_{m=1}^{n}a_{i_m}=$$
$$\sum_{m=1}^{n}a_{i_m}+c_{i_n}+\sum_{m=\mathcal{P}'(n)}^{N}b_{j_m}=$$
$$\sum_{m=1}^{\mathcal{P}(\mathcal{P}'(n))}a_{i_m}+c_{i_n}+\sum_{m=\mathcal{P}'(n)}^{N}b_{j_m} \quad (30)$$

由于 $\mathcal{P}'(n)\in[1,N]$，$\hat{Q}_n$ 也是 Q 中的一个元素。类似地，可证明 Q 中的任何元素都是 $\hat{Q}_n$ 中的元素。因此，$Q=\hat{Q}$ 且 $T_{2N}=\hat{T}_{2N}$，即调度策略的完成时间与其镜像调度时间相等。证毕。

参考文献：

[1] 彭安妮, 周威, 贾岩, 等. 物联网操作系统安全研究综述[J]. 通信学报, 2018, 39(3): 22-34.
PENG A N, ZHOU W, JIA Y, et al. Survey of the Internet of things operating system security[J]. Journal on Communications, 2018, 39(3): 22-34.

[2] 谢人超, 廉晓飞, 贾庆民, 等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155.
XIE R C, LIAN X F, JIA Q M, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 138-155.

[3] MCMAHAN H B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[J]. arXiv Preprint, arXiv: 1602.05629, 2016.

[4] LI M, ZHOU L, YANG Z C, et al. Parameter server for distributed machine learning[C]//Big Learning NIPS Workshop. Massachusetts: MIT Press, 2013: 1-10.

[5] KONEČNÝ J, MCMAHAN H B, RAMAGE D, et al. Federated optimization: distributed machine learning for on-device intelligence[J]. arXiv Preprint, arXiv: 1610.02527, 2016.

[6] WANG S Q, TUOR T, SALONIDIS T, et al. Adaptive federated learning in resource constrained edge computing systems[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1205-1221.

[7] MAO Y Y, YOU C S, ZHANG J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[8] ZHAO Y, LI M, LAI L, et al. Federated learning with non-IID data[J]. arXiv Preprint, arXiv: 1806.00582, 2018.

[9] XIE C, KOYEJO S, GUPTA I. Asynchronous federated optimization[J]. arXiv Preprint, arXiv: 1903.03934, 2019.

[10] CHEN Y J, NING Y, SLAWSKI M, et al. Asynchronous online federated learning for edge devices with non-IID data[C]//Proceedings of IEEE International Conference on Big Data (Big Data). Piscataway: IEEE Press, 2021: 15-24.

[11] CHAI Z, CHEN Y J, ZHAO L, et al. FedAT: a communication-efficient federated learning method with asynchronous tiers under Non-IID data[J]. arXiv Preprint, arXiv: 2010.05958, 2020.

[12] FEYZMAHDAVIAN H R, AYTEKIN A, JOHANSSON M. A delayed proximal gradient method with linear convergence rate[C]// Proceedings of 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). Piscataway: IEEE Press, 2014: 1-6.

[13] LUO B, LI X, WANG S Q, et al. Cost-effective federated learning in mobile edge networks[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(12): 3606-3621.

[14] 陶梅霞, 王栋, 孙瑞, 等. 联邦学习中基于时分多址接入的用户调度策略[J]. 通信学报, 2021, 42(6): 16-29.
TAO M X, WANG D, SUN R, et al. TDMA-based user scheduling policies for federated learning[J]. Journal on Communications, 2021, 42(6): 16-29.

[15] MA Q P, XU Y, XU H L, et al. FedSA: a semi-asynchronous federated learning mechanism in heterogeneous edge computing[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(12): 3654-3672.

[16] MORELLI M, KUO C C J, PUN M O. Synchronization techniques for orthogonal frequency division multiple access (OFDMA): a tutorial review[J]. Proceedings of the IEEE, 2007, 95(7): 1394-1427.

[17] NELSON R, KLEINROCK L. Spatial TDMA: a collision-free multihop channel access protocol[J]. IEEE Transactions on Communications, 1985, 33(9): 934-944.

[18] MO X P, XU J. Energy-efficient federated edge learning with joint communication and computation design[J]. Journal of Communications and Information Networks, 2021, 6(2): 110-124.

[19] CHEN M Z, POOR H V, SAAD W, et al. Convergence time optimization for federated learning over wireless networks[J]. IEEE Transactions on Wireless Communications, 2021, 20(4): 2457-2471.

[20] XU H L, CHEN M, MENG Z Y, et al. Decentralized machine learning through experience-driven method in edge networks[J]. IEEE Journal on Selected Areas in Communications, 2022, 40(2): 515-531.

[21] BLAZEWICZ J, ECKER K H, PESCH E, et al. Handbook on scheduling: from theory to practice[M]. Cham: Springer International Publishing, 2019.

[22] RAPPAPORT T. Wireless communications: principles and practice[J]. Microwave Journal, 2002, 45: 128-129.

[23] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[24] KRIZHEVSKY A. Learning multiple layers of features from tiny images[R]. 2009.

[25] CHAI Z, ALI A, ZAWAD S, et al. TiFL: a tier-based federated learning system[C]//Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing. New York: ACM Press, 2020: 125-136.

[作者简介]

马千飘（1993− ），男，贵州遵义人，博士，网络通信与安全紫金山实验室研究员，主要研究方向为边缘计算、联邦学习、分布式机器学习等。

贾庆民（1990− ），男，山东泰安人，博士，网络通信与安全紫金山实验室研究员，主要研究方向为算力网络、确定性网络、边缘智能、工业互联网等。

刘建春（1996− ），男，江苏盐城人，中国科学技术大学特任副研究员、硕士生导师，主要研究方向为物联网、边缘计算、分布式机器学习等。

徐宏力（1980− ），男，浙江宁波人，中国科学技术大学教授、博士生导师，主要研究方向为物联网、边缘计算、软件定义网络等。

谢人超（1984− ），男，福建南平人，博士，北京邮电大学副教授、硕士生导师，主要研究方向为信息中心网络、移动网络内容分发技术和移动边缘计算等。

黄韬（1980− ），男，重庆人，博士，北京邮电大学教授、博士生导师，主要研究方向为新型网络体系架构、内容分发网络、软件定义网络等。