Modelagem e Otimização do Aprendizado Federado para Justiça do Nível de Energia em Redes Sem Fio

Guilherme A. Thomaz[1] **e Miguel Elias M. Campista**[1]

Universidade Federal do Rio de Janeiro, GTA/DEL-Poli/PEE-COPPE, Brasil

{guiaraujo,miguel}@gta.ufrj.br

Abstract. *Federated learning distributes model training across the devices where data is generated. However, battery level transfer restricts training to the most efficient devices, reducing the data volume and, potentially, slowing the model convergence. This work proposes a new strategy that prevents some devices from having very low energy range levels, provided as an incentive to adhere to training. Thus, the server solves an optimization problem to adjust the epochs in which each device will be trained. An innovative contribution of the work is the reduction to a classic multichannel transmission problem, which is solved by a water-filling algorithm. Experimental results demonstrate a reduction in average energy consumption by up to 15% and in the standard deviation in the final battery level by up to 5.5% compared with FedAvg, PropEnerg, and PropEffic. The proposal increased the model accuracy by 8%, and the problem-solving time remains on the order of microseconds thanks to the linear complexity of the water-filling algorithm.*

Resumo. *O aprendizado federado distribui o treinamento de modelos entre os dispositivos em que os dados são gerados. Entretanto, a drenagem do nível da bateria restringe o treinamento aos dispositivos mais eficientes, reduzindo a quantidade de dados e, potencialmente, atrasando a convergência do modelo. Este trabalho propõe uma nova estratégia que evita que alguns dispositivos alcancem níveis de energia muito reduzidos, servindo como estímulo à adesão ao treinamento. Assim, o servidor resolve um problema de otimização para ajustar quantas épocas cada dispositivo irá treinar. Uma contribuição inovadora do trabalho é a redução para um problema clássico de transmissão multicanal, que é resolvido por um algoritmo de preenchimento com água. Os resultados experimentais demonstram uma redução no gasto médio de energia em até 15% e no desvio padrão no nível final de bateria em até 5,5% comparados com FedAvg, PropEnerg e PropEffic. A proposta elevou a acurácia do modelo em 8% e o tempo de solução do problema se mantém na ordem de microssegundos graças à complexidade linear do algoritmo de preenchimento com água.*

1. Introdução

Dispositivos em redes móveis coletam dados de sensores como câmera, microfone e GPS que revelam informações sensíveis. O envio desses dados para o treina-

Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001 e 88887.954253/2024-00, do CNPq (310234/2025-5, 407304/2025-8, 408255/2023-4 e 405940/2022-0), da FAPERJ (E-26/200.380/2023 e E-26/210.778/2025), da FAPESP (2023/00673-7 e 2023/00811-0) e da Fundação de Desenvolvimento da Pesquisa - Fundep - Rota 2030 em conjunto dos nossos parceiros Stellantis e Mobway.

mento de modelos de aprendizado de máquina pode revelar a localização do usuário ou conversas privadas. No aprendizado federado (*Federated Learning* – FL), um servidor distribui os parâmetros do modelo para que os nós da rede os treinem localmente [McMahan et al. 2017]. Entretanto, a drenagem da bateria durante o uso de um aplicativo pode causar insatisfação e desestímulo à participação entre usuários com dispositivos menos eficientes [Wang et al. 2023]. Cada usuário que deixa de treinar por causa da limitação energética representa uma redução na quantidade de dados usados no treinamento, induzindo vieses e desempenho insatisfatório [Marnissi et al. 2025]. Há uma lacuna na literatura de trabalhos que evitam a redução de energia de clientes com baixa disponibilidade energética ao incluí-los no treinamento.

Este trabalho otimiza a justiça (*fairness*) da energia entre os participantes do FL em uma rede móvel. O trabalho formula um problema, não discutido anteriormente na literatura, que visa alocar o número ótimo de épocas locais para cada dispositivo a fim de evitar que alguns deles fiquem com um nível de energia muito reduzido ao final de uma rodada. Para isso, uma proposta inovadora de redução do problema a um problema análogo de alocação de potência em canais digitais é adotada, permitindo o uso do algoritmo de preenchimento com água (*Water-Filling* – WF), com complexidade linear. Como a otimização de justiça em energia no FL é um problema inexplorado na literatura, foram propostas algumas métricas específicas para justiça de energia, bem como alguns *baselines* de comparação, como o *PropEnerg* e *PropEffic*. Os experimentos revelam que o WF, não só balanceia a energia residual entre os clientes, como também reduz o consumo total de energia. O tempo para solução do problema é desprezível em comparação ao tempo de uma rodada. Além disso, a técnica foi implementada no arcabouço Flower, e os resultados do treinamento de uma rede neural revelaram que a acurária aumentou em 8%.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 destaca a contribuição em relação aos trabalhos relacionados. A Seção 3 modela o problema em um ambiente de FL síncrono (clientes na mesma rodada) e centralizado (servidor agrega parâmetros recebidos no final da rodada). A Seção 4 deduz a equivalência com o problema de sistema multicanal, cuja solução por WF é descrita na Seção 5. A implementação da proposta é apresentada na Seção 6, enquanto as máquinas, modelos, métricas e *baselines* são apresentados na Seção 7. A Seção 8 apresenta e discute os resultados. Finalmente, a Seção 9 conclui este trabalho e apresenta os trabalhos futuros. A Tabela 1 enumera as principais variáveis utilizadas ao longo do trabalho.

2. Trabalhos Relacionados

Pesquisadores da Google propuseram o FL para aproveitar os recursos computacionais ociosos de dispositivos móveis para o treinamento de modelos [McMahan et al. 2017]. O servidor faz uma média ponderada dos parâmetros recebidos dos clientes (*FedAvg*), evitando a sobrecarga de rede com o envio de dados. Entretanto, a proposta original assume que apenas clientes conectados ao Wi-Fi e em carregamento participam, motivando trabalhos que modificam essa seleção de clientes. Por exemplo, o FedNorm-E seleciona os clientes que mais contribuirão para o treinamento, evitando o desgaste energético de dispositivos com baixa chance de contribuição [Zhao et al. 2022].

Diversos trabalhos discutem problemas de alocação de recursos no FL para oti-

mizar uma função objetivo, indo além do caso binário de seleção. Zhou *et al.* alocam largura de banda, potência de transmissão e frequência da CPU em uma rede de acesso por rádio, minimizando uma soma ponderada de energia e tempo. [Zhou et al. 2022]. Alguns trabalhos maximizam a justiça para garantir que os clientes não obtenham valores muito discrepantes de uma variável. O arcabouço FairEnergy seleciona a banda e a taxa de compressão de modo a minimizar a energia de comunicação, ao mesmo tempo em que garante a participação justa dos clientes ao longo das rodadas [Marnissi et al. 2025]. O conceito de justiça no nível de energia (*energy-fairness*) vem sendo discutido na área de redes sem fio por Gao *et al.* Os autores configuram parâmetros do LoRa para maximizar a menor energia entre os sensores (max-min) [Gao et al. 2019]. Isso evita que alguns nós morram antes dos outros, impedindo dados incompletos e topologias desconexas.

O algoritmo de WF, apesar de inicialmente voltado para alocação de potências em canais ortogonais, vem sendo usado em outros problemas na literatura. O artigo de Cui *et al.* é o único trabalho encontrado que combina o algoritmo de WF com a modelagem de energia em FL [Cui et al. 2024]. O objetivo dos autores é distribuir de forma justa uma recompensa entre os clientes, de forma que os que contribuem com mais energia para o treinamento sejam mais incentivados financeiramente.

O trabalho atual se diferencia dos anteriores nos seguintes aspectos:

- funciona tanto como um mecanismo de seleção de clientes quanto de alocação de recursos, diferente de [Zhao et al. 2022] no qual não há como dosar o nível de participação de um cliente;
- foca na justiça conforme o nível de energia no FL, ao invés de garantir justiça na quantidade de rodadas que cada cliente participa, conforme [Marnissi et al. 2025];
- realiza experimentos com clientes heterogêneos quanto à disponibilidade energética, em vez do cenário homogêneo discutido em [Cui et al. 2024];
- propõe uma otimização mais realista do que as propostas em [Zhou et al. 2022], [Marnissi et al. 2025] e [Cui et al. 2024], pois não depende do escalonamento de recursos de rádio ou frequências de *clock*;
- implementa a proposta no popular arcabouço Flower, utilizando parâmetros encontrados em especificações de dispositivos reais e parâmetros fáceis de serem estimados experimentalmente.

3. Modelagem do Problema

No cenário considerado, um servidor central distribui o treinamento de um modelo para I dispositivos sem fio. Cada dispositivo i, com $1 \leq i \leq I$, treina por x_i épocas. Em cada época, o dispositivo atualiza os parâmetros do modelo de acordo com seu conjunto de dados local, que é dividido em B_i lotes (*batches*). Para processar um lote, o dispositivo gasta c_i ciclos de *clock* da unidade de processamento, que pode ser uma CPU ou GPU. No caso de redes neurais, o processamento de um lote corresponde aos cálculos de retropropagação e descida do gradiente. Dependendo das técnicas de paralelismo no nível de arquitetura, *kernel* e bibliotecas, múltiplas amostras de um lote podem ser processadas concorrentemente, afetando o valor de c_i. No caso do treinamento de modelos mais simples, pode-se considerar cada amostra como um lote.

A partir de modelos da literatura [Zhou et al. 2022], o consumo de energia por época (ϵ_i) é:

$$\epsilon_i = B_i c_i f_i^2 \gamma_i, \tag{1}$$

Tabela 1. Principais notações.

Símbolo	Significado	Unid.	Símbolo	Significado	Unid.
I	Número de dispositivos		x_i	Épocas locais	
B_i	Número de lotes		T	Tempo da rodada	s
c_i	Ciclos de *clock* por lote		E_i^0, E_i	Energia inicial/final	J
f_i	Frequência de *clock*	Hz	δ	Número mínimo de épocas	
γ_i	Capacitância efetiva	Js	r_i	Redução no número de épocas	
ϵ_i	Energia por época	J	ψ	Número máximo de épocas	
τ_i	Tempo por época	s	N_i	CDE (Const. de Dispon. Ener.)	
$P_i^{\downarrow}$	Potência média consumida	W	k	Fração de épocas fixas	
$P_i^{\uparrow}$	Potência de carregamento	W	FQ	Fator de queda de energia médio	
η_i	Eficiência de carregamento		σ	Desvio padrão da energia	J
u_i, d_i	Tempo de *upload*/*download*	s	H	Entropia normalizada de seleção	

onde f_i é a frequência de *clock*, em Hz (Hertz), e γ_i é a capacitância efetiva, em $J \cdot s$ (Joule-segundo), que é um parâmetro relacionado com o gasto energético da tecnologia empregada pelo *hardware*. Já o tempo por época (τ_i), em s (segundo), é a razão entre o número total de ciclos de *clock* para processar todos os lotes e a frequência em ciclos por segundo, ou seja:

$$\tau_i = \frac{B_i c_i}{f_i}. \tag{2}$$

O *i-ésimo* dispositivo inicia a rodada com energia E_i^0 e a termina com energia E_i, ambas em J (Joule). A energia gasta devido às x_i épocas é $x_i \epsilon_i$. Ademais, a energia gasta pelo dispositivo ao longo da rodada devido a outros processamentos é $P_i^{\downarrow} T$ e a energia recebida pelo dispositivo devido a uma fonte de carregamento é $\eta P_i^{\uparrow} T$, onde $P_i^{\downarrow}$ é a potência média consumida por outras tarefas do dispositivo, $P_i^{\uparrow}$ é a potência média de carregamento, ambas em W (Watt), η é a eficiência do carregamento e T é o tempo da rodada, em s. Dessa forma, a energia de um dispositivo ao final de uma rodada é a energia inicial, menos o gasto de energia (consumida menos recebida), ou seja:

$$E_i = E_i^0 - x_i \epsilon_i - \left(P_i^{\downarrow} - \eta P_i^{\uparrow} \right) T. \tag{3}$$

Ao final de cada rodada, os dispositivos enviam seus parâmetros para o servidor agregá-los. O tempo para *upload* dos parâmetros pela rede sem fio é $u_i = (M/R_i^U) + t_i^U$ e o tempo para *download* dos parâmetros no início da rodada é $d_i = (M/R_i^D) + t_i^D$. Nessas equações, M é o tamanho do modelo, em Mb, R_i^U e R_i^D são as taxas do *up/down-link*, respectivamente, em Mb/s e t_i^U e t_i^D representam o impacto dos atrasos de propagação, processamento e fila no tempo de *up/download*, respectivamente, em s. O tempo, em s, que um dispositivo gasta para executar uma rodada (T_i) é a soma dos tempos de treinamento, *upload* e *download*, ou seja, $T_i = x_i \tau_i + u_i + d_i$. No FL síncrono, o tempo de uma rodada é $T = \max\ T_i$.

No início de cada rodada, o servidor resolve um problema de otimização para escolher por quantas épocas cada dispositivo i executará (x_i) de modo a maximizar a justiça do nível de energia dos I dispositivos ao final daquela rodada. Em problemas de justiça, é comum a definição de uma função utilidade, como a utilidade logarítmica [Cui et al. 2024]. Neste trabalho, a utilidade logarítmica de um dispositivo é uma função da sua energia final, ou seja, $U_i = \log E_i$. A utilidade total de uma solução,

utilizada como função objetivo do problema de otimização, é a soma:

$$U = \sum_{i=1}^{I} \log E_i. \tag{4}$$

A função $\log\ x_i$ é crescente, mas possui concavidade negativa, penalizando fortemente soluções nas quais algum x_i é pequeno, mesmo que x_j ($i \neq j$) seja grande. Outras funções utilidade poderiam ser utilizadas, mas, nesses casos, não seria possível realizar a simplificação proposta para utilizar o algoritmo de preenchimento com água.

O problema de otimização completo é:

$$\max_{\{x_i\}} \quad \sum_{i=1}^{I} \log E_i \tag{5}$$

$$\text{sujeito a:} \quad \sum_{i=1}^{I} x_i \geq \delta, \tag{6}$$

$$x_i \tau_i + u_i + d_i \leq T, \qquad \forall i \in [1, I], \tag{7}$$

$$x_i \in \mathbb{N}, \qquad \forall i \in [1, I]. \tag{8}$$

A primeira restrição define o valor mínimo de épocas, δ, que o servidor deve dividir entre os dispositivos, sendo δ um parâmetro constante. Essa restrição evita a solução trivial $x_i = 0, \forall i$. Dispositivos com f_i menor, apesar de mais eficientes energeticamente (Eq. 1), são mais lentos (Eq. 2). Assim, a segunda restrição limita o tempo da rodada, evitando soluções que tentam minimizar a energia apenas priorizando dispositivos muito lentos. A terceira restrição apenas impede soluções nas quais o número de épocas de um dispositivo é fracionário ou negativo. Uma redução no valor de x_i reduz a função objetivo sem violar a segunda restrição. Assim, a solução ótima satisfaz $\sum x_i = \delta$, pois qualquer aumento em $\sum x_i$ acima de δ reduz a utilidade, se afastando do ótimo. Consequentemente, o sinal "$\geq$" na primeira restrição pode ser substituído por "$=$".

4. Redução ao Problema de Comunicação Digital Multicanal

Esta seção demonstra que o problema de otimização proposto é matematicamente equivalente a um problema clássico de alocação de potência em sistemas de modulação multiportadora, que possui uma solução eficiente amplamente discutida na literatura [Haykin 2001].

A redução no número de épocas de um dispositivo (r_i) é definida como:

$$r_i \equiv \psi - x_i, \tag{9}$$

onde ψ é o número máximo de épocas que o servidor pode alocar para um dispositivo, ou seja, $\psi = \max x_i$. Pela restrição descrita na Eq. 7:

$$\psi = \max_i \frac{T - u_i - d_i}{\tau_i}. \tag{10}$$

Define-se, também de forma arbitrária, um parâmetro N_i, apelidado de constante de disponibilidade energética (CDE) do dispositivo i:

$$N_i \equiv \frac{E_i^0 - \epsilon_i \psi - \left(P_i^{\downarrow} - \eta P_i^{\uparrow}\right) T}{\epsilon_i}. \tag{11}$$

A sua interpretação física é a seguinte. *Suponha que o dispositivo i treine com o número máximo de épocas ($r_i = 0$ e $x_i = \psi$). O dispositivo precisa treinar por mais N_i épocas para drenar a sua energia restante para zero ($E_i = 0$).* As Equações 3, 9 e 11 são combinadas e manipuladas algebricamente para gerar a seguinte equação:

$$E_i = \epsilon_i N_i \left(1 + \frac{r_i}{N_i}\right). \tag{12}$$

Assim, a Eq. 5, referente à função objetivo do problema proposto, pode ser reescrita após as seguintes manipulações algébricas: i) mudança das variáveis de otimização de x_i para r_i, ii) substituição de E_i pela Eq. 12, e iii) aplicação da propriedade do logaritmo do produto (soma dos logaritmos). O resultado é:

$$\max_{\{r_i\}} \sum_{i=1}^{I} \left[\log\left(\epsilon_i N_i\right) + \log\left(1 + \frac{r_i}{N_i}\right)\right]. \tag{13}$$

Uma vez que a parcela $\sum \log\left(\epsilon_i N_i\right)$ é constante, ela não influencia na solução, podendo ser eliminada sem modificar a solução. Utilizando a Eq. 9, a restrição descrita na Eq. 6 pode ser reescrita da seguinte forma:

$$\sum_{i=1}^{I} r_i = \left(\sum_{i=1}^{I} \psi\right) - \delta \equiv R, \tag{14}$$

onde o novo parâmetro R é interpretado como *a máxima redução no número de épocas (somada entre todos os clientes)*.

Utilizando estes resultados (Eq. 13 sem a parcela constante e Eq. 14), o problema de otimização inicial pode ser reescrito na forma do seguinte problema reduzido:

$$\max_{\{r_i\}} \quad \sum_{i=1}^{I} \log\left(1 + \frac{r_i}{N_i}\right) \tag{15}$$

$$\text{sujeito a:} \quad \sum_{i=1}^{I} r_i = R, \tag{16}$$

$$r_i \in \mathbb{N}, \quad \forall i \in [1, I]. \tag{17}$$

Observe as seguintes deduções lógicas:

$$r_i \geq 0 \implies x_i \leq \psi \implies x_i \leq \left(T - u_i - d_i\right)/\tau_i.$$

A primeira sentença vem da Eq. 17. A segunda sentença combina a primeira sentença com a Eq. 9. A terceira sentença combina a segunda sentença com a Eq. 10. Dessa forma, é possível afirmar que a restrição dada pela Eq. 17 garante que a restrição descrita na Eq. 7 do problema original seja cumprida.

Este problema é análogo ao problema de distribuir a potência total, R, de um transmissor digital multiportadora entre I canais AWGN (*Additive White Gaussian Noise*) independentes e com mesma largura de banda e ganho (*path loss*). O *i-ésimo* canal tem

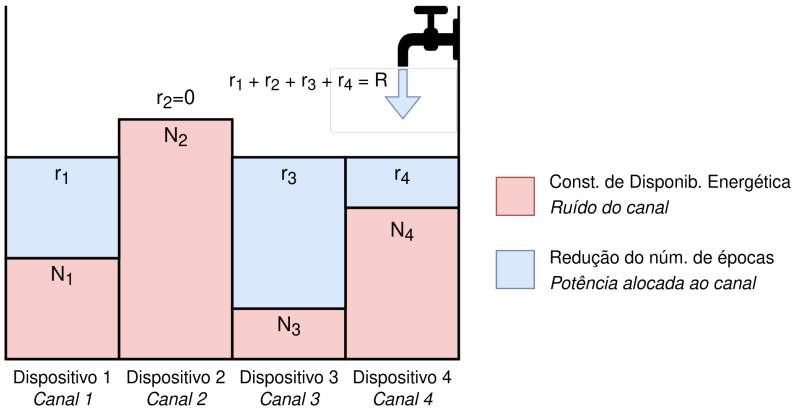


Figura 1. Algoritmo de preenchimento com água. Os textos em itálico descrevem a sua aplicação ao problema de alocação de potência em transmissores multicanal e os textos normais descrevem a aplicação ao problema proposto.

densidade espectral de potência do ruído naquela banda igual a N_i e a potência alocada para ele é r_i. A função objetivo a ser maximizada é proporcional à capacidade total dos canais pela fórmula de Shannon. A vantagem desta redução é a existência de algoritmos disponíveis na literatura para resolver esse problema de forma altamente eficiente.

A solução do problema reduzido é o vetor de reduções, $\mathbf{r} \in \mathbb{R}^I$, onde o *i-ésimo* elemento é r_i, enquanto que a solução do problema original é o vetor de épocas, $\mathbf{x} \in \mathbb{R}^I$, com elementos x_i. Usando a Eq. 9 e a definição de vetor coluna unitário $\mathbf{1}_I \in \mathbb{R}^I$, a solução do problema original é obtida a partir da solução do problema reduzido: $\mathbf{x} = \mathbf{1}_I\psi - \mathbf{r}$. Assim, o problema reduzido é um *proxy* para a solução do problema original proposto.

5. Algoritmo de Preenchimento com Água

Os métodos de Lagrange e KKT (*Karush–Kuhn–Tucker*) resolvem problemas genéricos de otimização convexa com restrições lineares a partir da solução de um sistema de equações lineares, o que pode ser complexo em problemas com muitas variáveis de otimização [Boyd and Vandenberghe 2004]. É provado que o problema de alocação de potência em sistemas multicanal é convexo. O algoritmo de preenchimento com água (*Water-Filling* – WF) resolve esse problema com complexidade linear no número de variáveis, evitando o uso de condições KKT. Os canais (dispositivos, i) são representados como tanques de mesma largura, mas com fundos irregulares, conforme a Figura 1. O volume total de água representa a potência total (redução de épocas total, R), enquanto que o volume de água em cada tanque é a potência alocada para aquele canal (redução de épocas de um dispositivo, r). Um tanque com fundo superior ao nível da água não é preenchido. O volume deve ser distribuído de modo a manter o mesmo nível de água em todos os tanques preenchidos. Uma variação desse algoritmo, que costuma encontrar soluções mais rapidamente, é chamado de preenchimento com água pelo método da busca bisseção (*Bisection Search Water-Filling* – BSWF). Esse método é uma versão modernizada do WF, cuja descrição é omitida por já ter sido abordada em alguns trabalhos da literatura [Xing et al. 2020].

Na presença de dispositivos com CDEs (Eq. 11) muito discrepantes, o problema

de otimização pode concentrar o número de épocas nos dispositivos mais eficientes, descartando o uso dos menos eficientes. Por um lado, isso funciona como um critério de seleção de clientes, que é uma etapa importante no aprendizado federado em larga escala. Por exemplo, o problema de otimização tende a selecionar dispositivos conectados a um carregador. Por outro lado, essa abordagem pode prejudicar a diversidade nos dados utilizados para treinamento e, consequentemente, o desempenho do modelo. Para mitigar esse problema, é proposta uma modificação no problema anterior. O servidor aloca $(1-k)\,\delta$ épocas usando o problema de otimização e aloca as $k\delta$ épocas restantes de forma uniforme entre os dispositivos, alocando ao todo δ épocas. Assim, o servidor executa os seguintes passos:

1. Soluciona o problema de otimização usando $(1-k)\,\delta$, com $0 \leq k \leq 1$, no lugar de δ, encontrando soluções x_i pelo BSWF.
2. Atribui a cada dispositivo um número de épocas $x_i' = x_i + \dfrac{k\delta}{I}$ no lugar de x_i.

Por exemplo, se o número de épocas a ser distribuído é $\delta = 100$, o número de dispositivos é $I = 10$ e o parâmetro $k = 0,5$, o servidor aloca metade do número de épocas uniformemente entre os clientes, enquanto que as outras 50 épocas são alocadas pelo preenchimento com água. Assim, cada cliente irá treinar por $5 + x_i$ épocas, onde o cliente com maior disponibilidade energética terá um valor maior de x_i. Para que x_i' seja a solução ótima quanto à justiça logarítmica no nível de energia, conforme proposto pelo problema de otimização, deve-se escolher $k = 0$. Um aumento em k tende a aumentar a diversidade entre os clientes participantes, uma vez que até mesmo os clientes com baixa disponibilidade energética precisarão executar algumas rodadas de treinamento. Em compensação, essa abordagem tende a prejudicar a otimização da energia. Com $k = 1$, o problema reduz-se à abordagem tradicional do FL, sem ciência da energia, na qual os clientes treinam pelo mesmo número de rodadas ($x_i' = \delta/I$ e $x_i = 0$).

6. Implementação

Esta seção discute como os principais desafios para incorporar a solução em um *framework* prático de FL foram superados. Um dos desafios diz respeito à obtenção dos parâmetros R e N_i em um cenário com dispositivos reais. Para isso, propõe-se um procedimento de estimação de parâmetros em uma rodada preliminar. Nela, cada dispositivo treina por um número fixo de épocas (x_A) e verifica a variação de energia, em J, (ΔE_A), e o tempo de execução da rodada, em s ($T_{i,A}$). Em seguida, ele treina por mais x_B épocas ($x_B \neq x_A$) e obtém a variação de energia ΔE_B e o tempo $T_{i,B}$. Pela Eq. 12:

$$\begin{cases} \Delta E_A = -x_A \epsilon_i - P_i T_{i,A} \\ \Delta E_B = -x_B \epsilon_i - P_i T_{i,B} \end{cases}$$

com $P_i = P_i^{\downarrow} - \eta P_i^{\uparrow}$. A solução do sistema permite estimar ϵ_i e P_i de forma trivial. Devem ser utilizados valores pequenos de x_A e x_B para evitar com que o consumo de potência do dispositivo mude ao longo da estimação. O dispositivo também estima o tempo por época: $\tau_i = T_{i,A}/x_A = T_{i,B}/x_B$. A medição de energia em J, necessária para o procedimento de estimação, é obtida medindo o estado de carga (*State-of-Charge* – SoC), em % de bateria ($SoC_{\%}$), e aplicando a fórmula de conversão $E = SoC_{\%} \cdot C_{mAh} \cdot 10^{-3} \cdot 3600 \cdot U$, onde C_{mAh} é a capacidade da bateria em mAh (mili-Ampere-hora) e U é a tensão da

bateria em V (Volt). Essas são especificações que costumam ser conhecidas em sistemas comerciais. Em cenários nos quais o valor de SoC reportado pelo dispositivo é não-linear, essa equação pode ser facilmente substituída por uma outra relação, mas essa discussão foge do escopo desse trabalho, que assume que o valor obtido de E é suficientemente exato. O dispositivo pode utilizar temporizadores para medir, diretamente, o tempo para baixar o modelo no início (d_i) e para enviar o modelo ao final (u_i) da rodada preliminar. Uma vez que são conhecidos os valores de E_i^0, ϵ_i, $P_i^{\downarrow} - \eta P_i^{\uparrow}$, τ_i, u_i e d_i de cada dispositivo, o servidor consegue calcular ψ (Eq. 10) e, em seguida, usá-lo para calcular R (Eq. 14) e N_i (Eq. 11). Vale lembrar que o tempo máximo por rodada (T), o número mínimo de épocas total (δ) e o parâmetro de diversidade (k) são parâmetros do problema de otimização, ajustados na criação do treinamento federado no servidor.

O *framework* de aprendizado federado selecionado para implementar a estratégia é o Flower, que permite sobrescrever métodos da classe que implementa o servidor de agregação e o cliente de treinamento. Na implementação proposta, a primeira rodada do cliente é a rodada preliminar, com 10 épocas. Ao final de cada rodada, o cliente envia ao servidor um relatório no formato JSON (*Javascript Object Notation*) com suas características e os parâmetros do modelo local. Após executar o FedAvg para gerar o modelo global, o servidor lê os relatórios dos clientes, calcula os parâmetros e resolve o problema de otimização, programado em Python. O número de épocas que cada cliente treinará é enviado em um relatório em JSON, junto com o modelo global, para iniciar a próxima rodada.

7. Cenário Experimental

Esta seção apresenta as métricas e parâmetros utilizados nos dois experimentos. A máquina utilizada possui uma CPU i9-10900 e 32GB de memória RAM. Os resultados são apresentados com intervalo de confiança de 95%.

7.1. Cenário do Experimento 1

O Experimento 1 (Exp. 1) tem como objetivo avaliar individualmente o desempenho do algoritmo em comparação com outras estratégias de definição do número de rodadas usadas como base:

1. *FedAvg*: o número total de rodadas (δ) é dividido igualmente entre os I dispositivos, ou seja, o número de épocas do cliente i é dado por $x_i = \delta/I$. Essa é a estratégia típica do FL, que não é ciente da justiça de energia.

Tabela 2. Faixas de valores dentro das quais os parâmetros de cada cliente são sorteados. Em ambos os experimentos, $P_i^{\downarrow} = 0$, $P_i^{\uparrow} = 0$, $U = 3,7V$ e $M = 4MB$.

Parâmetro de um cliente	Símbolo	Faixa de valores (Exp. 1)	Faixa de valores (Exp. 2)
Capacidade em mAh	C_{mAh}	$[1900, 2000]$	$[1900, 2000]$
Estado de carga inicial em $\%$	$SoC_{\%}$	$[10, 40]$	$[10, 40]$
Número de lotes	B	$[90, 110]$	$[10, 20]$
Capacitância efetiva em Js	γ	$[50 \cdot 10^{-28}, 55 \cdot 10^{-28}]$	$[50 \cdot 10^{-38}, 55 \cdot 10^{-38}]$
Taxa de *upload* em Mbps	t^U	$[40, 60]$	$[40, 60]$
Taxa de *download* em Mbps	t^D	$[40, 60]$	$[40, 60]$
Frequência de *clock* em GHz	f	$[2.8, 3.2]$	$[2.8, 3.2]$
Ciclos de *clock* por lote	c	$[24 \cdot 10^6, 36 \cdot 10^6]$	$[24 \cdot 10^{10}, 36 \cdot 10^{10}]$

2. *PropEnerg*: o número de épocas de um cliente é diretamente proporcional ao seu nível de energia no início da rodada, ou seja, $x_i = \delta \cdot \frac{E_i^0}{\sum_{j=1}^{I} E_j^0}$.
3. *PropEffic*: o número de épocas de um cliente é inversamente proporcional ao consumo de potência por época (ϵ_i/τ_i), ou seja, $x_i = \delta \cdot \frac{\tau_i/\epsilon_i}{\sum_{j=1}^{I} \tau_j/\epsilon_j}$.

Apesar das últimas duas estratégias serem cientes de energia, elas se diferenciam da proposta por não haver solução formal de um problema de otimização.

Para esse primeiro experimento, um *script* simula $I = 20$ clientes que irão executar uma rodada de aprendizado federado, com tempo máximo de $T = 120s$. O número de épocas a ser alocado pela estratégia é dado por $\delta = 100$. Uma execução do experimento começa com a geração aleatória dos parâmetros dos clientes, seguindo valores realistas. Cada parâmetro é sorteado conforme uma distribuição uniforme dentro de um intervalo, conforme indicado na Tabela 2, na qual as faixas de valores são inspiradas em dispositivos reais. Em seguida, o número de épocas de cada cliente (x_i) é definido por cada uma das três estratégias usadas como base e mais a estratégia proposta (baseada em WF). Para cada um desses quatro casos, o *script* calcula a energia final (E_i) de cada dispositivo e o tempo da rodada.

Assim, o Experimento 1 abstrai a tarefa de aprendizado em si e foca apenas em avaliar a proposta em termos das seguintes figuras de mérito:

- Queda percentual no nível de energia de um cliente após uma rodada (fator de queda de energia, $FQ^{(i)}$):

  $$FQ_i = 1 - \frac{E_0^{(i)}}{E^{(i)}},$$

  Para avaliar a estratégia, considera-se a média entre os clientes ($FQ = \sum_{j=1}^{I} FQ_i/I$). Quanto menor esta métrica, maior a capacidade da estratégia de economizar energia dos dispositivos.
- Desvio padrão do nível de energia entre os clientes ao final de uma rodada (σ):

  $$\sigma = \sqrt{\frac{1}{I-1}\sum_{i=1}^{I}\left(E_i - \overline{E}\right)^2}, \text{com } \overline{E} = \frac{1}{I}\sum_{i=1}^{I} E_i$$

  Essa métrica mede o quão distintos são os níveis de energia dos dispositivos ao final da rodada, ou seja, espera-se que estratégias que priorizam a justiça do nível de energia reduzam seu valor.
- Tempo para definição do número de rodadas, seja pelo problema de otimização proposto ou pelas estratégias usadas como base. Mede a eficiência computacional da estratégia proposta.
- Tempo da rodada (T).

7.2. Cenário do Experimento 2

Já o Experimento 2 (Exp. 2) avalia o impacto da estratégia baseada em WF na convergência de um modelo federado após 10 rodadas globais, com $\delta = 100$ épocas distribuídas entre os clientes no início de cada rodada. A tarefa consiste na classificação

Tabela 3. Parâmetros de treinamento.

Nome do parâmetro	Valor do parêmetro
Camadas	784 (ReLU), 200 (ReLU), 10 (linear)
Função e Método de Otimização	Entropia Cruzada. SGD (*Stochastic Gradient Descent*) com $\eta = 0.05$
Amostras de por cliente	900 (treino) + 100 (validação)

de dígitos no conjunto de dados MNIST, dividido igualmente entre $I = 10$ dispositivos, usando um perceptron multicamadas (*Multilayer Perceptron* – MLP) implementado no PyTorch. Os parâmetros dos clientes são sorteados nas faixas apresentadas na última coluna da Tabela 2. Informações detalhadas da configuração experimental são apresentadas na Tabela 3. A taxa de aprendizado e o número de dados por cliente foram propositalmente reduzidos para evitar a convergência do modelo em poucas rodadas. O experimento é repetido para $k = 0$ (otimizador aloca 100 épocas), $k = 1$ (cada cliente treina por 10 épocas) e $k = 0,5$ (50 épocas são distribuídas igualmente entre os clientes e 50 épocas são alocadas pelo otimizador).

Para avaliar como o otimizador distribui as épocas entre os clientes neste experimento, é definida uma métrica chamada entropia normalizada de seleção:

$$H = -\frac{1}{\log I} \sum_{i=1}^{I} \frac{x_i}{\delta} \log\left(\frac{x_i}{\delta}\right). \tag{18}$$

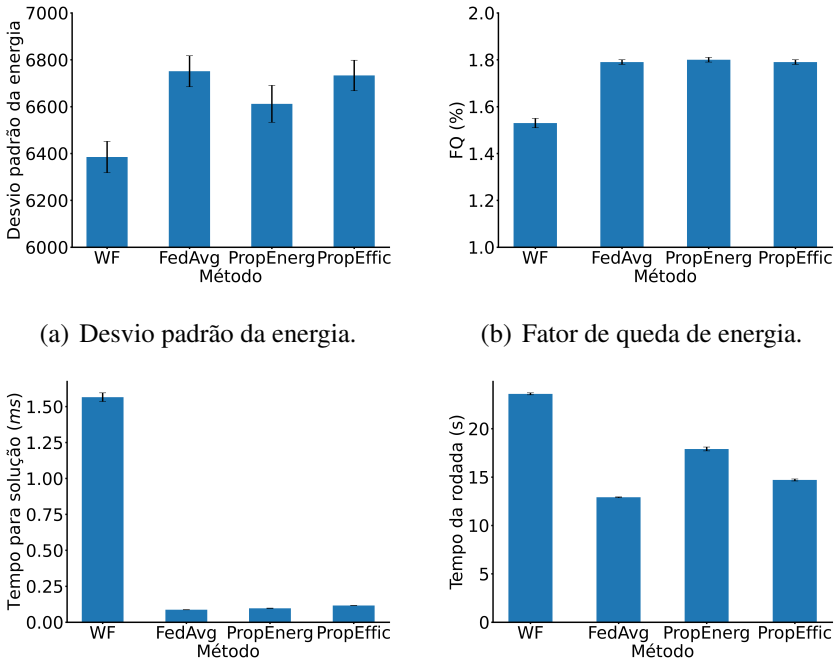
Utilizando a teoria da informação (Lema 2.1.1 e Teorema 2.6.4 de [Cover and Thomas 2006]), é possível provar que $0 \leq H \leq 1$, sendo 1 se, e somente se, a distribuição de épocas for uniforme ($x_i = \delta/I, \forall i$) e 0 se, e somente se, apenas um cliente treinar ($\exists i = c$ tal que $x_c = \delta$ e $x_j = 0, \forall j \neq c$). Assim, essa métrica é tão mais próxima de 1 quanto mais distribuídas forem as épocas em uma rodada.

8. Resultados

Esta seção discute os resultados dos experimentos descritos na Seção 7. Os códigos se encontram disponíveis no GitHub, no link: `https://github.com/GTA-UFRJ/OptimizingFairnessRemainingBatteryFL`. No mesmo link, é possível obter os artefatos do artigo, na forma de um arquivo `README.md` que documenta os requisitos e detalha os passos para reprodução dos experimentos.

8.1. Resultados do Experimento 1

A Figura 2 ilustra os resultados do Experimento 1. O método baseado em preenchimento de água (WF) é o único que produz uma redução no desvio padrão da energia em relação ao FedAvg estatisticamente significativa, conforme ilustra a Figura 2(a). As variações cientes de energia (PropEnerg e ProfEffic) apresentam resultados cujos intervalos de confiança se sobrepõem ao do FedAvg. Isso ocorre, pois o WF é o único método que otimiza a justiça no nível de energia explicitamente. Essa pode ser a diferença entre um cliente ser capaz ou não de terminar uma tarefa federada. Vale ressaltar que a tradução desse resultado em porcentagem de bateria depende, de forma estocástica, dos parâmetros dos dispositivos. A Figura 2(b) mostra que o WF também é o único que apresenta um fator de queda de energia médio (FQ) menor do que o dos métodos sem otimização, que



(a) Desvio padrão da energia.

(b) Fator de queda de energia.

(c) Tempo da otimização.

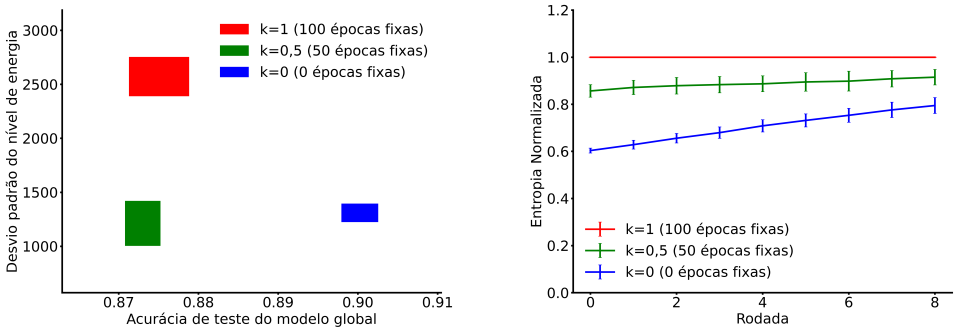
(d) Tempo da rodada.

Figura 2. Resultados do Experimento 1.

apresentam resultados estatisticamente compatíveis. Assim, a proposta não reduz apenas a desigualdade entre o nível de bateria dos clientes, uma vez que os clientes são beneficiados com um consumo médio de energia 5,5% menor. Apesar de o tempo para alocar e calcular o número de rodadas de cada cliente ser muito maior para o WF, a Figura 2(c) ilustra que a solução ocorre em pouco mais de 1,5ms, que é um tempo muito menor do que a duração da rodada, definido no máximo em 25s. Isso evidencia que a solução do problema por WF não introduz um consumo de recursos computacionais elevado, impedindo que o problema proposto prejudique o consumo de energia dos dispositivos, o que iria contra a proposta original. Por fim, a Figura 2(d) revela que o WF é o único método que aloca as épocas de forma a explorar todo o tempo máximo da rodada. Isso ocorre porque o principal fator que aumenta a eficiência energética de um dispositivo é a frequência de *clock*, conforme Equação 1. Como consequência, o otimizador prioriza clientes mais lentos. Apesar de isso ter levado a um tempo de rodada maior nesse experimento, a proposta se beneficia do fato de que o valor de tempo da rodada tem um comportamento determinístico e é escolhido pelo desenvolvedor. O parâmetro T pode ser reduzido para que a rodada atenda a uma restrição de tempo menor. Nesse caso, o otimizador será obrigado a alocar mais rodadas para dispositivos com maior *clock*, o que pode reduzir a justiça final.

8.2. Resultados do Experimento 2

A Figura 3(a) ilustra os resultados do treinamento do modelo no Experimento 2. A base e a altura de cada retângulo representam os intervalos de confiança na acurácia final do modelo e no desvio padrão da energia final, respectivamente. Alocar 100 épocas locais igualmente entre os 10 clientes (retângulo vermelho) leva a um resultado com maior desvio padrão de energia comparado com o resultado obtido com a alocação das épocas



(a) Variação da energia e acurácia.

(b) Entropia normalizada de seleção (H).

Figura 3. Resultados do Experimento 2.

pelo otimizador (retângulo azul). Isso confirma que a otimização do ambiente federado para justiça na energia cumpre a sua função, tornando o nível de energia mais homogêneo ao final. Quando o otimizador aloca apenas metade das épocas disponíveis (retângulo verde), a redução no desvio padrão de energia é compatível com o cenário no qual o otimizador aloca todas as épocas (retângulo azul).

Em todos os cenários, o modelo convergiu para um valor de acurácia que permanece praticamente constante. A acurácia foi consistentemente superior ao utilizar o método de otimização para alocar todas as épocas. Nesse cenário, o otimizador funciona como um mecanismo de seleção de clientes, ou seja, a cada nova rodada, o problema de otimização descarta dispositivos com baixa disponibilidade energética (conforme será ilustrado a seguir, na Figura 3(b)). Diferente dos outros dois casos ($k = 0,5$ e $k = 1$) nos quais todos os clientes são selecionados para treinar, quando $k = 0$, o otimizador seleciona apenas alguns clientes. O FedAvg não apresenta bom desempenho quando modelos com pesos heterogêneos são agregados, de forma que muitos trabalhos na literatura explicitamente limitam o número de modelos a serem agregados [Long et al. 2023]. Essa redução no número de modelos sendo agregados pelo FedAvg a cada rodada leva a uma melhor convergência, conforme indicado pelos resultados.

A Figura 3(b) ilustra a evolução da entropia normalizada de seleção (Eq. 18) ao longo das rodadas. O FedAvg tradicional divide as épocas igualmente, produzindo H máximo (curva em vermelho). O otimizador concentra as épocas nos clientes com maior disponibilidade energética, reduzindo H (curva em azul). Isso faz com que a energia desses clientes reduza rapidamente nas primeiras rodadas, de modo que, nas rodadas seguintes, o otimizador comece a alocar as épocas de maneira mais uniforme. Isso é uma consequência do fato de que o problema de otimização busca, por definição, uniformizar a disponibilidade energética entre os clientes. A curva em verde representa o caso intermediário, no qual metade das rodadas são fixas entre os clientes.

9. Conclusões

Este trabalho propõe o algoritmo de preenchimento com água como uma forma rápida de definir o número de épocas locais dos clientes do aprendizado federado, visando reduzir a discrepância no nível de energia entre os clientes. As principais contribuições são: i) a transposição da modelagem teórica para a implementação prática usando

especificações e ferramentas comerciais, ii) a redução tanto na média quanto na variância do consumo de energia entre os clientes, e iii) a possibilidade do uso da proposta como um mecanismo de seleção de clientes que mantém elevada acurácia. A adequação para cenários descentralizados, a implementação em *smartphones* e a experimentação com tarefas mais complexas estão previstas para trabalhos futuros.

Referências

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*, chapter 5, pages 215–288. Cambridge University Press, Cambridge.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley-Interscience, 2nd edition.

Cui, Y., Tong, W., Liu, T., Cao, K., Zhou, J., Xu, M., and Wei, T. (2024). Energy-Aware Incentive Mechanism for Hierarchical Federated Learning Using Water Filling Technique. *IEEE Transactions on Industrial Informatics*, 20(12):14214–14225.

Gao, W., Du, W., Zhao, Z., Min, G., and Singhal, M. (2019). Towards Energy-Fairness in LoRa Networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 788–798.

Haykin, S. (2001). *Communication Systems*, chapter 6, pages 431–439. John Wiley & Sons, Inc., New York, 4th edition. Section 6.12: Multichannel Modulation.

Long, G., Xie, M., Shen, T., Zhou, T., Wang, X., and Jiang, J. (2023). Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26(1):481–500.

Marnissi, O., Hammouti, H. E., and Bergou, E. H. (2025). FairEnergy: Contribution-Based Fairness meets Energy Efficiency in Federated Learning. *arXiv preprint arXiv:2511.15454*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Wang, E., Chen, B., Chowdhury, M., Kannan, A., and Liang, F. (2023). Flint: A platform for federated learning integration. *Proceedings of Machine Learning and Systems*, 5:21–34.

Xing, C., Jing, Y., Wang, S., Ma, S., and Poor, H. V. (2020). New viewpoint and algorithms for water-filling solutions in wireless communications. *IEEE Transactions on Signal Processing*, 68:1618–1634.

Zhao, J., Feng, Y., Chang, X., and Liu, C. H. (2022). Energy-efficient client selection in federated learning with heterogeneous data on edge. *Peer-to-Peer Networking and Applications*, 15(2):1139–1151.

Zhou, X., Zhao, J., Han, H., and Guet, C. (2022). Joint Optimization of Energy Consumption and Completion Time in Federated Learning. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1005–1017.