Agente K-alibra: Estratégia para Seleção de K-Clientes em Aprendizado Federado autônoma

Rafael O. Jarczewski[1], Eduardo Cerqueira[2], Antonio A. F. Loureiro[3], Leandro A. Villas[1], Allan M. de Souza[1]

[1]Universidade Estadual de Campinas, Brasil
[2]Universidade Federal do Pará, Brasil
[3]Universidade Federal de Minas Gerais, Brasil

r200219@dac.unicamp.br, cerqueira@ufpa.br,
loureiro@dcc.ufmg.br, {lvillas, allanms}@unicamp.br

Abstract. *O Aprendizado Federado (FL) estabeleceu-se como uma abordagem robusta para o treinamento distribuído de modelos, preservando a privacidade dos dados. Contudo, FL ainda sofre de problemas de escalabilidade e sobrecarga de rede. Uma das estratégias de mitigação pode ser através da seleção dinâmica de clientes, porém, a maioria dos trabalhos foca qualitativamente em quais clientes selecionar e não considera o número de clientes selecionados. Essa rigidez impede que o sistema se adapte adequadamente à dinâmica temporal do sistema distribuído, resultando em altos números de clientes ou poucos clientes de modo ineficiente. Essa falha é ocasionada pela dificuldade de algoritmos estáticos em adaptar-se dinamicamente às necessidades do sistema. Para mitigar essa rigidez, apresentamos o K-Agent, um orquestrador baseado em Language Models (LM) que ajusta dinamicamente o número de clientes via um ciclo de percepção, raciocínio e ação. Experimentos demonstram que o agente equilibra custos e estabilidade, reduzindo o tráfego de dados entre 44,4% e 59% comparado à literatura, validando a eficácia de agentes na otimização de hiperparâmetros em FL.*

Resumo. *Federated Learning (FL) has established itself as a robust approach for distributed model training while preserving data privacy. However, FL still suffers from scalability and network overhead issues. One mitigation strategy could be through dynamic client selection; however, most work focuses qualitatively on which clients to select and does not consider the number of clients selected. This rigidity prevents the system from adequately adapting to the temporal dynamics of the distributed system, resulting in high client numbers or inefficiently low client numbers. This flaw is caused by the difficulty of static algorithms in dynamically adapting to the system's needs. To mitigate this rigidity, we present K-Agent, a Language Models (LM)-based orchestrator that dynamically adjusts the number of clients via a cycle of perception, reasoning, and action. Experiments demonstrate that the agent balances costs and stability, reducing data traffic by 44.4% to 59% compared to the literature, validating the effectiveness of agents in hyperparameter optimization in FL.*

1. Introdução

O Federated Learning (FL) emergiu como um paradigma para o treinamento colaborativo de modelos de Machine Learning (ML) e Inteligência Artificial (IA) entre múltiplos

clientes conectados [McMahan et al. 2017], onde os modelos são treinados localmente e apenas os pesos do modelo são compartilhados com o servidor central para produzir um modelo global. Isso permite um melhor aprendizado ao aproveitar mais recursos computacionais e dados na borda, sem comprometer a privacidade dos participantes enviando dados brutos para um servidor central [McMahan et al. 2017].

Um dos principais desafios do FL é o *overhead* de comunicação [Kairouz et al. 2021], onde o compartilhamento modelos sobrecarrega a rede e prejudica a adoção de FL em escala. Para mitigar os impactos do *overhead*, soluções de seleção eficiente de clientes, compressão e poda de modelos e quantidade de rodadas necessárias para convergência foram propostas [Fu et al. 2023]. Neste contexto, percebese que o *overhead* no cenário federado ainda é um problema de pesquisa em aberto [Kairouz et al. 2021].

Métodos de seleção de clientes focam no *trade-off* entre poder computacional e qualidade dos dados (utilidade e necessidade) de cada cliente em cada rodada, selecionando grupos de clientes mais importantes para o treinamento, reduzindo o *overhead* e evitando custos computacionais desnecessários [de Souza et al. 2024, Maciel et al. 2024, Jarczewski et al. 2024]. Técnicas otimizadas de agregação de modelos mitigam oscilações e ruídos no modelo global causados por dados heterogêneos [Nguyen et al. 2024, Jarczewski et al. 2026], reduzindo a quantidade de rodadas necessárias para o treinamento e, consequentemente, o *overhead* de comunicação, enquanto abordagens de personalização de modelos adaptam o modelo global para atender às necessidades específicas de cada cliente [Piao et al. 2024, Talasso et al. 2025]. Outras técnicas visam através de técnicas de pruning do modelo, reduzindo a precisão dos pesos, para aliviar os custos de treinamento e comunicação, poda do modelo, que remove parâmetros redundantes [Shah and Lau 2023].

No entanto, a complexidade e a heterogeneidade inerentes aos ambientes distribuídos de FL tornam as soluções algorítmicas isoladas insuficientes para melhorar consistentemente o desempenho e a eficiência do sistema. Devido à complexidade multifatorial do sistema distribuído, estratégias de seleção frequentemente consideram apenas poucos critérios para tomar a decisão de qual cliente selecionar. Negligenciando contextualização individual e global do sistema federado, critérios esses que podem mudar durante o treinamento, tornando-o ainda mais complexo.

A inadequação de abordagens estáticas é corroborada por Fu et al. [Fu et al. 2023], que destacam a dificuldade crítica em identificar a quantidade ótima de clientes (K) a ser selecionada em cada iteração de treinamento. Definir o valor K possibilita a convergência do modelo mais rápida e a redução dos custos de comunicação sem sobrecarregar os dispositivos. Contudo, o dinamismo de ambientes de FL, por exemplo, como encontrados em cidades inteligentes, exige uma capacidade de adaptação que métodos tradicionais raramente alcançam.

Para endereçar essa necessidade de orquestração inteligente, surgem os Agentes de Modelos de Linguagem (Agente-LM) capazes de planejar, raciocinar e agir em cenários complexos [Wang et al. 2024]. A arquitetura de sistemas de agentes compreende quatro módulos [Acharya et al. 2025]. *i) Perfil:* estipula os objetivos e comportamentos do agente, bem como suas restrições, limitações e habilidades acessíveis via técnicas de *Prompt Engeneering (PE)*[1] (e.g. Chain-of-Thought (CoT) e Few-Shot Prompting (FWP))

[1]Prompt Engineering é uma estratégia de manipulação da entrada de texto dos Language Model (LM)

[Wei et al. 2022, Brown et al. 2020]. *ii) Memória:* armazena o conhecimento contextual de curto e longo prazo, propiciando uma tomada de decisão consistente e evolutiva. *iii) Planejamento:* segmenta tarefas complexas em subtarefas executáveis e raciocina sobre estratégias para eleger a melhor decisão conforme o estado do ambiente atual. *iv) Ação:* componente responsável pela execução das decisões mediante a interação com o ambiente, outros agentes ou ferramentas externas.

A integração dessa arquitetura em sistemas de FL permite que *frameworks* como ReAct [Yao et al. 2022] estruturem a resolução de problemas aliando deliberação e execução. Em vez de operar sob parâmetros fixos, o agente utiliza seus mecanismos de raciocínio e memória para modelar a heterogeneidade do sistema e interpretar metadados contextuais. Essa autonomia é o que viabiliza a superação do desafio proposto por Fu et al. [Fu et al. 2023], onde o agente passa a definir dinamicamente o valor de K, ajustando o volume de clientes participantes com base no estado atual da federação e refinando estratégias de agregação em tempo real.

Dessa maneira, visando superar essas limitações estáticas e preencher essa lacuna de adaptabilidade na seleção de clientes, o presente trabalho apresenta o K-Agent, um agente baseado em LM que ajusta dinamicamente o número de clientes para cada rodada de comunicação no FL baseado na análise de metadados individuais dos clientes e informações globais da federação. Para isso, K-Agent utiliza (*In-context Learning*) e relaciona as informações coletadas dos clientes com uma visão macro da federação e, através disso, define K dinamicamente em cada rodada de comunicação. As contribuições deste artigo podem ser sumarizadas da seguinte forma:

- K-Agent: Introdução de um agente que define a quantidade de clientes de forma dinâmica e contextual para FL.
- Análise extensiva sobre custo e eficiência da abordagem proposta, explorando diferentes cenários e técnicas de *prompt engeneering*.
- Uma implementação pública de código aberto integrando um Agente-LM em um ambiente federado promovendo a ciência aberta e reprodutibilidade[2].

O restante deste artigo é organizado da seguinte forma: A Seção 2 apresenta os trabalhos relacionados, destacando suas vantagens e desvantagens. A Seção 3 introduz a solução proposta. A Seção 4 discute os resultados obtidos. Por fim, a Seção 5 apresenta a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

A seleção de clientes é um componente crítico para a eficiência do FL. A abordagem padrão, introduzida pelo algoritmo FedAvg [McMahan et al. 2017], utiliza uma seleção aleatória de uma fração fixa de clientes a cada rodada. Essa estratégia não considera a heterogeneidade do sistema e dos dados, resultando frequentemente em problemas de convergência lenta e gargalos causados por dispositivos lentos (*stragglers*). Para superar essas limitações, foram propostos métodos de seleção que consideram outros critérios ao escolher os conjuntos de clientes para o treinamento. Nesta seção, classificamos os trabalhos em quatro categorias principais: *i)* estratégias baseadas em recursos de *hardware*; *ii)* abordagens híbridas focadas na utilidade dos dados; e *iii)* estratégias de quantidade de clientes dinâmica.

para guiar a linha de raciocínio que deve ser seguida pelo modelo.

[2]https://github.com/rafaeloj/sbrc2026/

Seleção Orientada a Recursos de Hardware. O foco desta categoria é mitigar o impacto da heterogeneidade de dispositivos, priorizando a viabilidade da execução dentro de restrições temporais ou energéticas. Nishio et al. [Nishio and Yonetani 2019] propuseram o protocolo FedCS, que trata a seleção como um problema de otimização de recursos. O método introduz uma etapa de aquisição de informações onde os clientes reportam suas capacidades (CPU, canal, tamanho da base de dados). O servidor então executa um algoritmo guloso para selecionar o maior número possível de participantes que consigam completar o treino dentro de um limite de tempo estimado. Contudo, o uso de um *threshold* rígido pode excluir dispositivos capazes, mas ligeiramente mais lentos, negligenciando a importância estatística de seus dados. Visando a eficiência energética, Xu et al. [Xu and Wang 2021] desenvolveram o *framework* OCEAN. O método utiliza o algoritmo Set Expansion Algorithm (SEA) para alocação sequencial, priorizando clientes com base em um equilíbrio entre deficiência energética e qualidade do canal. O OCEAN poupa dispositivos para rodadas futuras, mas assume implicitamente uma disponibilidade garantida, o que é arriscado em cenários de alta intermitência onde a contribuição de um cliente deve ser capturada assim que ele estiver disponível.

Seleção Híbrida e Baseada na Utilidade dos Dados. Esta vertente busca acelerar a convergência priorizando clientes cujos dados oferecem maior contribuição ao modelo global, frequentemente medida pelo erro de treinamento ou gradientes. O *framework* Oort, proposto por Fan et al. [Lai et al. 2021], seleciona clientes que maximizam a utilidade estatística (alto erro de treinamento) sob restrições de tempo. Utilizando um algoritmo Multi-Armed Bandit (MAB), o Oort penaliza clientes lentos para evitar gargalos. No entanto, sua eficácia diminui em ambientes altamente dinâmicos, onde a constante reexploração do MAB pode gerar sobrecarga excessiva frente à rápida mudança nas condições da rede. Além de necessitar da calibragem de hiperparâmetros como o fator de exploração e definição da quantidade de clientes selecionados. Similarmente, Cho et al. [Jee Cho et al. 2022] introduziram o *Power-of-Choice (PoC)*, uma estratégia de amostragem enviesada. O método prioriza clientes com maior perda local, sob a premissa de que eles possuem informações valiosas para o modelo. Apesar de acelerar a convergência em testes controlados, o PoC falha ao desconsiderar a latência desses nós. Frequentemente, dispositivos com maior erro são também os que possuem conexões mais instáveis, e sua seleção prioritária pode degradar o tempo total da rodada. Além de exigir um número fixo de clientes por rodada.

Estratégias de Quantidade de Clientes Dinâmica. Diferentemente dos trabalhos anteriores que focam em *quais* clientes selecionar (mantendo o número K fixo ou limitado apenas pelo hardware), poucas abordagens investigam a adaptação do número de clientes necessários para a convergência. Um avanço relevante nessa direção é o DEEV, proposto por de Souza et al. [Souza et al. 2023]. O DEEV ajusta dinamicamente a quantidade de clientes selecionados baseando-se no desempenho do modelo global. A estratégia aplica uma função de decaimento que reduz progressivamente o K à medida que o modelo amadurece, evitando computação redundante nas etapas finais. Embora demonstre redução significativa de custos (até 90%), a abordagem assume que a redução da quantidade de clientes é sempre benéfica no final e depende da transmissão constante de métricas de desempenho, não prevendo cenários onde o aumento temporário do K possa ser necessário para escapar de mínimos locais ou lidar com dados Non-Identical Independent Distributed (Non-IID) severos.

A Tabela 1 sumariza as características dos trabalhos discutidos em quatro grupos, quantidade de clientes dinâmica (K), se levam em consideração restrições de hardware,

a performance do modelo e se visam otimizar o valor de K. Observa-se que, embora a literatura tenha avançado na seleção qualitativa (quem selecionar), a definição quantitativa (quantos selecionar) ainda é tratada majoritariamente como um hiperparâmetro fixo ou uma consequência passiva das limitações de hardware. A proposta de um método dinâmico para a seleção do K, capaz de se adaptar à variabilidade do ambiente e do aprendizado, permanece como uma lacuna aberta que este trabalho visa preencher.

Tabela 1. Comparativo de Estratégias de Seleção de Clientes

Método	**Abordagem**	**Quantidade de Clientes Dinâmica**	**Restrição de Hardware**	**Estado do Aprendizado**	**Otimização do K**
FedAvg	Random Fixed	–	–	–	–
FedCS	Greedy (Max)	✓	✓	–	–
OCEAN	Seq. Allocation	✓	✓	–	–
Oort	MAB (Utility)	–	✓	✓	–
PoC	Biased Sampling	–	–	✓	–
DEEV	Decay Function	✓	–	✓	✓
K-Agent	**Active K-Search**	✓	✓	✓	✓

✓: Considera o critério; –: Fixo/Limitado.

3. K-Agent: Método agêntico para definição do número de clientes a cada rodada de comunicação

K-Agent otimiza dinamicamente a quantidade de clientes (K) para seleção em sistemas de FL. Diferente de abordagens tradicionais, o K-Agent atua como um orquestrador capaz de interpretar o estado semântico e estatístico do treinamento, utilizando ferramentas desenvolvidas e descrições das heurísticas da literatura (p.e., PoC, DEEV e OCEAN) para selecionar os clientes. K-Agent opera em um ciclo de percepção, raciocínio e ação, fundamentado em técnicas de CoT que descrevem a sequência de passos lógicos que devem ser seguidos pelo modelo de LM e aprendizado de contexto via FWP que introduz exemplos de práticos de quais critérios devem ser considerados para definição do K.

3.1. Definição do Problema

Seja $\mathcal{C} = \{1, \ldots, N\}$ o conjunto total de clientes disponíveis na federação, onde cada $i \in \mathcal{C}$ possui um conjunto de dados local d_i. O objetivo global é aprender um modelo sobre o conjunto de dados particionado $\mathcal{D} \bigcup_{i \in \mathcal{C}} d_i$. O protocolo de treinamento opera em rodadas discretas $t \in \{1, \ldots, T\}$. Os clientes selecionados ($S \subseteq C$) recebem o modelo global w_g^t e realizam o treinamento em seus dados d_i. Então, cada cliente i retorna as atualizações do modelo local w_i para o servidor. No final da rodada t, o modelo global é atualizado através da média ponderada das atualizações enviadas pelos clientes de acordo com a eq. (1).

$$w_g^{t+1} = \sum_{i=1}^{|S|} \frac{|d_i|}{|D|} w_i^t \tag{1}$$

Dessa forma, o objetivo do FL é minimizar a função objetivo $F(w_g)$:

$$\min_w F(w_g) \quad \text{onde} \quad F(w_g) = \sum_{i=1}^{|C|} \frac{|d_i|}{|\mathcal{D}|} \mathcal{L}(w_g; x_i, y_i) \tag{2}$$

onde w_g é o modelo global, $\mathcal{L}$ corresponde à função de perda, x_i e y_i, respectivamente, são entradas e rótulos do conjunto de dados d_i para cada cliente. K-Agente tenta encontrar a cardinalidade do conjunto S^t.

O K-Agent endereça três desafios: *i)* Como integrar as informações da federação do modo eficiente no agente para mitigar as alucinações de raciocínio? *ii)* Como fornecer o contexto histórico das rodadas anteriores para proporcionar um comportamento evolutivo? *iii)* Como permitir que o LM interaja com o sistema e consulte metadados dos clientes (p.e., nível bateria, tempo de treinamento, etc) atuando como parte do orquestrador?

Para superar os obstáculos e desafios da proposta, adaptamos os módulos da arquitetura de um agente descritos na Seção 1 da seguinte forma: o módulo de Planejamento utiliza ferramentas de recuperação para ancorar o raciocínio nos metadados reais dos clientes *(i)*. Já o módulo de memória é estruturado para indexar *(logs)* de rodadas anteriores e metadados dos clientes *(ii)*, permitindo que o módulo de ação execute e interaja diretamente com o orquestrador *(iii)*. Essa integração garante que o agente interaja de modo preciso, baseando suas decisões em dados concretos e mitigando lacunas de informação.

3.2. Visão Geral

K-Agent é implementado no servidor central e é acionado no início de cada rodada de comunicação t e é composto por três módulos interconectados: Plano, Memória e Ação, conforme ilustrado na Figura 1. A primeira etapa do processo (***1***) consiste na definição do algoritmo de seleção que será utilizado durante o treinamento. A definição é realizada antes de iniciar o treinamento e o mesmo algoritmo é utilizado durante todas as rodadas. Na etapa seguinte (***2***), a cada rodada de comunicação o agente é acionado para definir o valor ideal para K, que manipula sob os metadados dos clientes e da federação destacados na etapa (***3***). Nesta parte as ferramentas permitem que o agente obtenha conhecimento sobre as informações de treinamento de cada cliente (p.e., tempo de treinamento, desempenho do modelo, qualidade do link, etc.) e da federação como média global de cada atributo. Em seguida, na etapa (***4***), o agente injeta o hiperparâmetro no algoritmo definido e o sistema executa a solução. O servidor, então, na etapa ***(5)***, envia o modelo global para os clientes selecionados, dando continuidade ao fluxo tradicional do FL.

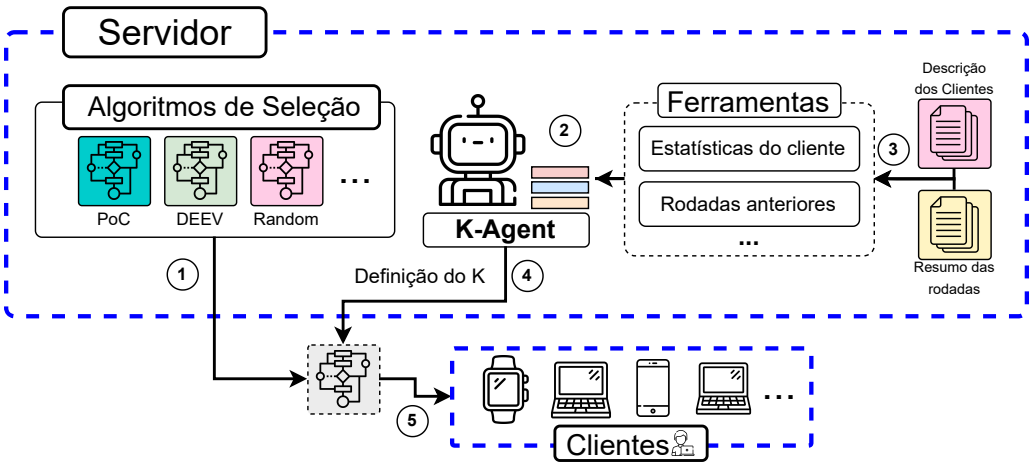


Figura 1. Visão geral do K-Agent.

Com esses módulos o agente raciocina, planeja e interage com o ambiente de FL. O K-Agent toma decisões baseadas no contexto construído, o qual integra conhecimento estático e dinâmico. Para mitigar alucinações e garantir decisões mais eficientes, o *system Prompt*[3] do agente é inicializado com um resumo de estratégias validadas na literatura. Assim, o agente recebe descrições das soluções, tais como a lógica de restrição de *hardware* do FedCS ou a estratégia de decaimento de cardinalidade do *DEEV*. Dessa forma, o Large Language Model (LLM) não opera de forma aleatória, mas sim como um orquestrador que seleciona qual heurística melhor se aplica ao cenário atual. Por exemplo, ao detectar um platô de convergência, o agente pode optar por reduzir K para economizar recursos, citando a lógica do DEEV como justificativa; alternativamente, em fases iniciais de alta divergência, pode reduzir K para estabilizar o gradiente global igual descoberto por Xu et al. [Xu and Wang 2021]. A adaptação temporal do agente é viabilizada através de uma memória de curto prazo injetada no contexto via FWP e CoT [Wei et al. 2022] conhecidas como técnicas de *PE*.

3.2.1. Módulos do Agente e Algoritmo

K-Agent é composto por três módulos principais: planejamento, memória e de ferramentas e as soluções para os desafios de implementação do K-Agent. Em seguida, descrevemos como é o fluxo através da descrição algorítmica.

Módulo de Planejamento: agrega e traduz os dados brutos da federação, como métricas de desempenho do sistema e características dos clientes, em uma representação textual estruturada compreensível pelo LM criando a *descrição dos clientes* e o *resumo das rodadas anteriores*. Além disso, processa o contexto atual utilizando CoT para definir o valor de K. Para viabilizar interação com o ambiente e coleta de informações numéricas e estatísticas em descrições textuais p_i sem exceder a janela de contexto[4] Em cenários com muitos clientes, utilizamos métodos de resumo e recuperação. Neste trabalho, optamos por permitir que o agente acesse essas informações sob demanda, garantindo um planejamento mais assertivo baseado nas observações do ambiente.

Módulo de Memória: atua como um registro histórico que armazena tuplas de (Estado, Ação, Resultado), representando respectivamente as descrições dos clientes, resumo da rodada, as decisões do modelo LM e as respostas das ferramentas. Isso permite que o agente avalie o impacto de suas decisões anteriores (e.g., se a seleção de K na rodada $t-1$ prejudicou a convergência). A memória é estruturada em tuplas $h_t = (M_t, K_t, \Delta acc_t, \Delta loss_t)$, onde M_t representa as métricas e estatísticas dos clientes na rodada t, K_t indica quais dispositivos foram selecionados e, por fim, Δacc_t e $\Delta loss_t$ representam o ganho ou a perda de acurácia e *loss* resultantes da seleção realizada na rodada anterior.

Módulo de Ação: Interage com o ambiente, executa tarefas e recupera informações. Foram desenvolvidas funções auxiliares para impedir que todos os dados p_i sejam passados diretamente ao modelo, assim evitando sobrecarga. O LM utiliza ferramentas de recuperação de dados (`get_client_info` ou `get_client_attributes`) para adquirir conhecimentos contextuais individuais. Já para adquirir conhecimento global usa ferramentas como (e.g. `get_summary_round`, `get_client_stat`) que retor-

[3] *System Prompt* é uma outra forma de definir o módulo de perfil conforme apresentado na Seção 1

[4] Janela de contexto refere-se ao limite de *tokens* que uma LM consegue processar simultaneamente durante a inferência.

nam resultados com métricas estatísticas gerais. Ao finalizar seu processo de raciocínio, utiliza ferramentas de ação para agir no ambiente (e.g. `set_k` e `select_clients`). Dessa forma o agente consegue interagir com o ambiente de forma eficiente e realizar escolhas autônomas.

Algorithm 1 Algoritmo de Seleção Dinâmica via K-Agent

Require: Base de Conhecimento B_t, Janela de Memória H, Modelo Inicial w_0
1: $H_0 \leftarrow \emptyset$ {Inicializa memória vazia}
2: **for** $t = 1$ **to** T **do**
3: **Fase 1: Percepção do Estado**
4: $\mathcal{C} \leftarrow$ GetActiveClients()
5: $M \leftarrow$ MeasureState($\mathcal{C}$) {Métricas: Loss, Acurácia, Tempo, etc.}
6: **Fase 2: Construção do Prompt e Inferência**
7: $P \leftarrow$ Concat(B_t, H_{t-1}, M)
8: $S_t \leftarrow$ LLM.Generate(P) {Define K e utiliza ferramenta de seleção}
9: **Fase 3: Execução e Agregação**
10: $w_t, Acc_t, Loss_t \leftarrow$ FedAvg($w_{t-1}, \mathcal{S}_t$)
11: **Fase 4: Feedback e Atualização de Memória**
12: $\Delta acc \leftarrow Acc_t - Acc_{t-1}$
13: $\Delta loss \leftarrow Loss_t - Loss_{t-1}$
14: $h_t \leftarrow \{M, K_t, \Delta acc, \Delta loss\}$
15: $H_t \leftarrow H_{t-1} \cup h_t$
16: **end for**

O procedimento operacional do K-Agent é detalhado no Algoritmo 1. O fluxo inicia-se com a inicialização das estruturas de memória e segue um ciclo iterativo por T rodadas. Na Fase 1 (linhas 4–5), o sistema realiza a percepção do ambiente, identificando os clientes ativos e computando o vetor de métricas de estado M, que inclui dados de desempenho e latência. Em seguida, na Fase 2 (linhas 7–8), o agente recebe o prompt contextualizado com a base de conhecimento B_t e o histórico H_{t-1}, gerando a decisão de seleção S_t via inferência do LM. A efetivação dessa decisão ocorre na Fase 3 (linha 10), onde o servidor executa a agregação FedAvg com o subconjunto definido. Por fim, a Fase 4 (linhas 12-15) fecha o ciclo de controle: o sistema calcula os diferenciais de acurácia (Δacc) e perda ($\Delta loss$) resultantes da rodada, encapsulando essas informações junto com a ação tomada em uma nova entrada de memória h_t, que é agregada ao histórico H para refinar as decisões futuras.

4. Análise de Desempenho

Esta seção avalia o K-Agent considerando diferentes cenários pontuando vantagens e desvantagens, bem como os efeitos dos diferentes métodos de construção de Agente-LM. Na Seção 4.1 detalhamos o cenário e a base de dados utilizados, já na Seção 4.2 descrevemos como implementamos o K-Agent, os *baselines* de comparação e também as as métricas utilizadas. Por fim, a Seção 4.3 aborda a análise dos experimentos realizados.

4.1. Cenário de Simulação e Base de Dados

A infraestrutura de FL foi implementada com o framework *Flower* por conta da sua orquestração de clientes. Para a tarefa de classificação de imagens, adotamos o dataset CIFAR-10, composto por 60.000 imagens coloridas de 32×32 pixels, distribuídas em 10 classes e o dataset MNIST, composto por 70.000 imagens em tons de cinza de 28×28 pixels, distribuídas em 10 classes.

Para simular um cenário realista e Non-IID, o particionamento dos dados entre os clientes seguiu uma distribuição de Dirichlet com parâmetro $\alpha = 0.1$. Esta configuração induz um desbalanceamento severo, em que cada cliente detém amostras de poucas classes, forçando o algoritmo de seleção a buscar estrategicamente uma composição de clientes que estabilize o modelo global. O cenário experimental foi configurado com 50 clientes, treinados ao longo de 50 rodadas de comunicação. Cada experimento foi executado três vezes, sendo reportada a média dos resultados.

4.2. Implementação do K-Agent, Modelo, Baselines e Métricas

Para implementação do K-Agent utilizamos o *framework LangGraph* visto que permite gerenciar o fluxo de raciocínio e memória (grafo de estados), integrado à ferramenta *Ollama* para a inferência local dos LLM. O modelo de aprendizado utilizado pelos clientes consiste em uma CNN composta por duas camadas convolucionais seguidas por três camadas densas (400, 120 e 84 neurônios) uma vez que são problemas de classificação de imagens. O treinamento local utilizou o otimizador *Stochastic Gradient Descent (SGD)* e a função de perda *Cross Entropy*. Para aferir o desempenho do método proposto, comparamos o K-Agent contra quatro estratégias de seleção consolidadas na literatura: Seleção Aleatória, Round Robin, estratégia determinística que seleciona clientes sequencialmente, PoC [Jee Cho et al. 2022] e Oort [Lai et al. 2021]. Para o LM do agente testamos três modelos diferentes: qwen3:8b (MQ8b), llama3.2:3b (MLL3b) e llama3.1:8b (MLL8b)

A eficácia do K-Agent foi investigada através das seguintes métricas: *(i)***Acurácia distribuída:** Desempenho do modelo nos dados de teste dos dispositivos. O objetivo é maximizar a acurácia com menor número de rodadas; *(ii)* **Tempo de Seleção (TS):** Tempo total de processamento para definir a quantidade e a seleção dos clientes. Para o K-Agent, somam-se o tempo de inferência do LM e o tempo de execução do algoritmo de seleção invocado; e *(iii)* **Dados transmitidos (DT):** Sobrecarga de rede gerada pela solução, calculada com base no tamanho do modelo (em bytes) e na quantidade acumulada de clientes selecionados durante a federação.

4.3. Análise dos Resultados

Foram conduzidos três conjuntos de experimentos que investigam: i) **Impacto do LM:** como a escolha do modelo afeta a decisão do valor de K; ii) **Engenharia de Prompt:** influência de técnicas como CoT, FWP e Apenas Descrição (AD) da tarefa que deve ser resolvida[5]; iii) **Desempenho Comparativo:** K-Agent *vs.* Literatura (com $K = \{5, 10, 15\}$ fixos e K aleatório); e iv) **Análise Semântica:** avaliação qualitativa das decisões do agente.

4.3.1. Impacto do Modelo de Linguagem e Prompt

A Tabela 2 apresenta a acurácia, o tempo de seleção (TS) e o K-médio resultante de diferentes combinações de modelos e prompts para o MNIST e CIFAR10. Observa-se que o modelo *MQ8b* tendeu a escolher valores de K menores, porém com maior desvio padrão. Essa variabilidade permitiu atingir acurácias superiores, porém com TS superiores. Enquanto para o MNIST podemos perceber que o *MLL3b* atingiu resultados similares aos

[5]Para mais informações, veja os detalhes de implementação no repositório do artigo. `https://github.com/rafaeloj/sbrc2026/`

outros modelos com um TS inferior devido a cadeias de pensamentos mais curtos e objetivas. Contudo, apesar disso, em alguns casos o agente se perde no raciocínio sendo necessárias outras chamadas aumentando seu tempo como demonstrado no Oort usando CoT.[6]

Tabela 2. Impacto do LLM e Prompt na Seleção e Desempenho

Cifar 10										
Modelos	**Prompt**	**PoC**			**Aleatório**			**Oort**		
		K Médio	**Acurácia** (%)	**TS** (s)	**K Médio**	**Acurácia** (%)	**TS** (s)	**K Médio**	**Acurácia** (%)	**TS** (s)
MQ8b	**AD**	7±5	38±13	3,79	6±5	**39±11**	4,71	12±6	**39±14**	4,45
	FWP	6±4	38 ±13	1,43	5±4	38±15	2,28	9±5	38±15	2,96
	CoT	7±5	**39±13**	1,79	5±4	37±14	1,65	9 ±4	39 ±13	2,53
MLL8b	**AD**	10±0	37±13	**0,18**	10±0	38±10	0,38	9±2	38±13	**0,19**
	FWP	8±6	37±13	3,21	10±7	38±12	1,88	8±4	38+15	3,92
	CoT	8±2	38 ±12	0,53	8±2	37±14	0,51	8 ±1	39 ±12	0,57
MLL3b	**AD**	10±0	36±13	0,29	9±1	36±14	0,21	9±1	39±11	0,31
	FWP	5±0	35±13	0,31	10±7	36±11	1,88	9±1	38±14	0,27
	CoT	9±1	35+15	0,20	8±2	36±13	**0,15**	8±1	38±13	0,22
MNIST										
Modelos	**Prompt**	**PoC**			**Aleatório**			**Oort**		
		K Médio	**Acurácia** (%)	**TS** (s)	**K Médio**	**Acurácia** (%)	**TS** (s)	**K Médio**	**Acurácia** (%)	**TS** (s)
MQ8b	**AD**	7±6	95,97±3	4,00	6±5	**96,90±1**	8,65	11±6	**97,06±1**	4,45
	FWP	7±5	96,68±2	1,67	5±4	**96,32±2**	1,74	8±6	**97,22±1**	1,97
	CoT	7±5	**96,50±2**	1,63	8±2	95,25±2	0,18	9±6	**97,05±1**	2,21
MLL8b	**AD**	10±0	**96,04±3**	**0,17**	10±2	**96,66±2**	0,18	9±2	**97,05±1**	0,19
	FWP	10±1	95,76±2	0,16	6±1	95,86±3	0,34	9±1	**97,21±1**	0,33
	CoT	8±2	**96,64±2**	0,26	8±2	95,89±2	0,21	5±1	**97,09±1**	0,28
MLL3b	**AD**	10±0	95±3	0,31	9±1	**96,64±1**	0,20	9±1	**97,22±1**	0,29
	FWP	9±2	**96,26±2**	**0,16**	7±2	**96,13±2**	**0,14**	8±1	**97,00±1**	**0,16**
	CoT	9±1	**96,13±2**	0,19	8±2	**96,25±2**	0,18	5±1	**96,77±1**	2,21

Apenas Descrição (AD), Few-shot Prompting (FWP), Chain-of-Thought (GoT), Tempo de Seleção (TS)

Em relação às técnicas de prompt, o CoT apresentou TS majoritariamente menor. Algumas exceções ocorreram com o *MLL8b*, onde a descrição simples foi mais rápida, porém, essa configuração resultou em valores de K estáticos para PoC e Aleatório, reduzindo a eficácia adaptativa do agente. A Figura 3 detalha o overhead de tempo, evidenciando que o *FWP* (laranja) demandou mais tempo de processamento devido à necessidade de correção frequente de chamadas de ferramentas pelo agente, especialmente no modelo *MLL3b* (barras lisas), que sofreu com erros de alucinação na AD.

4.3.2. Eficiência: K-Agent versus Baselines

A Tabela 3 compara o desempenho do K-Agent (utilizando definição dinâmica de K) contra métodos com K fixo ou aleatório. Os resultados demonstram que o K-Agent atinge acurácia similar ou superior aos métodos da literatura, porém com uma redução significativa nos DT.

Esta eficiência é corroborada pela Figura 2, que ilustra a economia de rede comparada ao método randômico para definição do K. Por exemplo, ao utilizar o modelo *MQ8b*

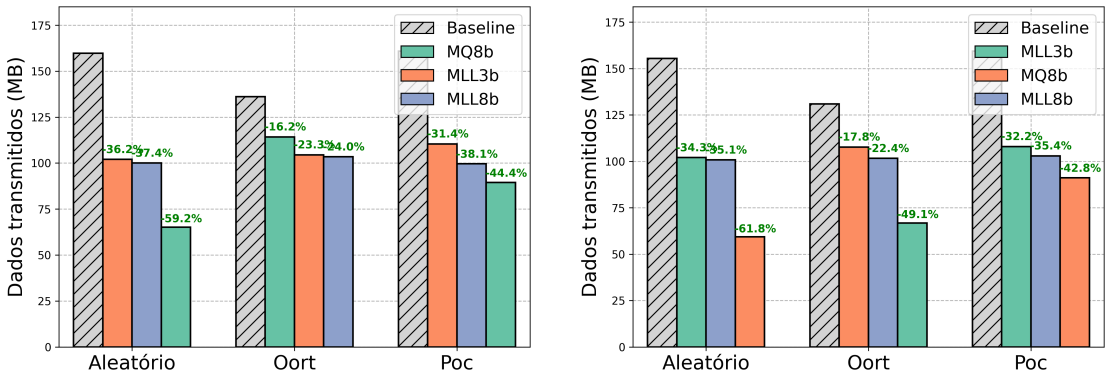
[6]CoT beneficia modelos grandes o suficiente para generalizar linhas de raciocínio *step-by-step* como evidenciado por Wei et al. [Wei et al. 2022].

Tabela 3. Comparação de Desempenho: K-Agent vs. Literatura

Seleção	Métrica	Definição K									
		Cifar10					MNIST				
		5	**10**	**15**	**Aleatório**	**K-Agent**	**5**	**10**	**15**	**Aleatório**	**K-Agent**
Oort	**DT** (MB)	57	109	160	136	103	58	112	161	130	101
	Acurácia (%)	36±12	38±11	38±14	37±14	**39 ±12**	96,96±1	96,94±1	**97,29±1**	**97,29±1**	97,09±1
PoC	**DT** (MB)	58	117	176	160	89	58	117	176	159	91
	Acurácia (%)	33±13	37±10	38±13	36±13	**39±13**	**96,25±1**	**96,24±2**	**96,28±2**	**96,21±3**	**96,50±2**
Aleatório	**DT** (MB)	58	117	176	159	65	58	117	176	155	102
	Acurácia (%)	35±14	37±12	**38±12**	37±14	37±14	95,84±2	**96,37±2**	**96,49±2**	**96,36±2**	**96,25±2**
RRobin	**DT** (MB)	58	117	176	161	**92**	58	117	176	162	78
	Acurácia (%)	33±15	**37±9**	**37±9**	36±13	**37±12**	92,21±9	91,94±10	94,74±5	**96,18±2**	**96±2**

Dados Transmitidos

com a estratégia PoC, obteve-se uma redução de 44,4% no tráfego de dados mantendo acurácia superior. Em comparação com a seleção randômica, a redução chegou a 59% mantendo a mesma acurácia.



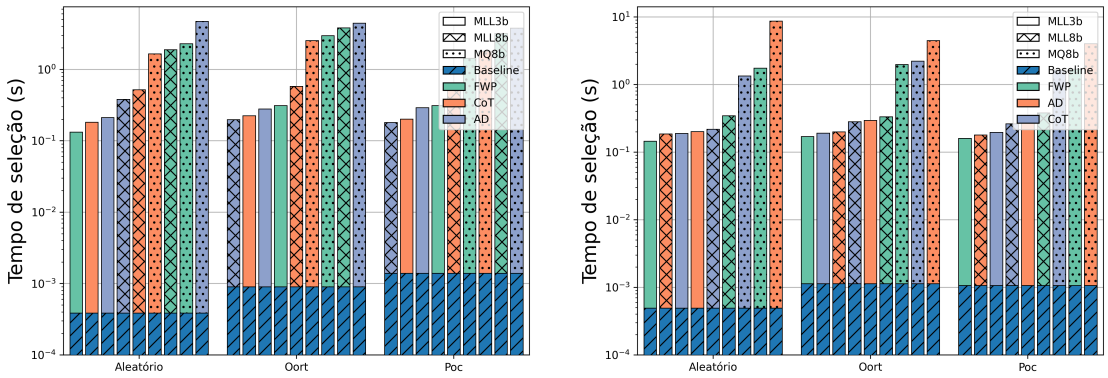
(a) CIFAR10 (b) MNIST

Figura 2. Volume de dados e ganho percentual em relação à definição randômica de K.

4.3.3. Análise Qualitativa e Convergência

A Figura 4 ilustra que a superioridade do K-Agent transcende o desempenho métrico, fundamentando-se na sua capacidade de adaptação contextual. Ao divergir da estratégia fixa na rodada 25, o agente demonstra uma percepção dinâmica do estado da federação que métodos estáticos não possuem. No momento em que o sistema detecta uma iminente instabilidade, a permanência do valor de $K = 5$ atua como um mecanismo de controle de variância, estabilizando o gradiente global, antes de retomar uma exploração mais agressiva de clientes após a rodada 30. Essa transição suave entre conservadorismo e exploração é o que garante não apenas uma convergência mais rápida, mas uma robustez operacional superior em ambientes heterogêneos.

Complementarmente, a arquitetura baseada em agentes introduz uma camada de auditabilidade e explicabilidade. Por meio da análise dos logs de raciocínio, é possível decodificar a lógica causal por trás de cada alteração no valor de K, transparente. Essa



(a) CIFAR10 (b) MNIST

Figura 3. Overhead no tempo de seleção para diferentes prompts.

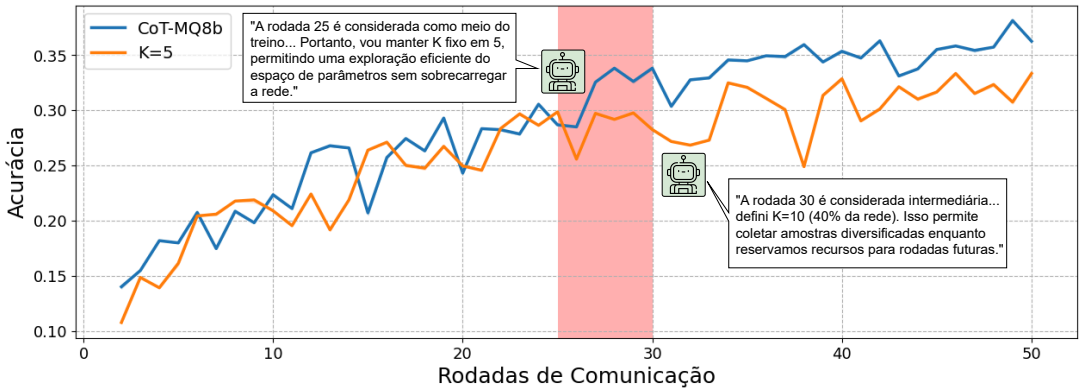


Figura 4. Convergência do PoC: O intervalo destaca a adaptação dinâmica da estratégia do agente.

rastreabilidade permite que desenvolvedores auditem as heurísticas do modelo em tempo real, validando se a escolha por maior diversidade de clientes em rodadas avançadas foi motivada pela estabilidade do modelo ou por métricas de desempenho específicas. Assim, o K-Agent estabelece um novo paradigma onde a eficácia do treinamento é indissociável da clareza sobre o comportamento do sistema.

5. Conclusão

Este trabalho apresentou o K-Agent, uma abordagem agêntica baseada em Agente-LM para a orquestração dinâmica da cardinalidade de clientes (K) em ambientes de FL. Diferentemente das estratégias tradicionais que tratam K como um hiperparâmetro fixo ou dependente apenas de restrições de hardware, o mecanismo proposto demonstrou capacidade de adaptar a seleção de participantes com base na dinâmica temporal de convergência e nas métricas do sistema.

Os resultados experimentais, conduzidos num cenário Non-IID com o dataset CIFAR-10, validaram a eficácia da abordagem. O K-Agent atingiu uma acurácia comparável ou superior aos métodos do estado da arte, como Oort e PoC, mas com uma eficiência de comunicação significativamente maior. Destaca-se a redução de até 59% nos dados transmitidos em comparação com a seleção aleatória e 44,4% em relação ao

PoC, mantendo a estabilidade do modelo global.

A análise qualitativa do fluxo de raciocínio do agente confirmou que a utilização de Chain-of-Thought e memória histórica permitiu decisões coerentes, em que o agente optou por explorar o espaço de parâmetros no início do treino e conservar recursos em fases de estabilização. Isso corrobora a hipótese de que agentes baseados em LM podem atuar como orquestradores autônomos eficazes, superando a rigidez de heurísticas estáticas.

Como trabalhos futuros, pretendemos investigar a escalabilidade do K-Agent em cenários com maior heterogeneidade de dispositivos e avaliar o impacto de técnicas de quantização no próprio LM para reduzir o tempo de inferência no servidor, mitigando o overhead observado em alguns modelos. Além de explorar alternativas como o próprio Agente decidir quais clientes devem participar do treinamento.

Disponibilidade de Artefatos

Em linha com princípios e práticas de Ciência Aberta, disponibilizamos a implementação da nossa abordagem através do repositório: `https://github.com/rafaeloj/sbrc2026/`.

Acknowledgments

Este projeto foi apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado Arquitetura Cognitiva (Fase 3), DOU 01245.003479/2024-10. Este trabalho foi parcialmente patrocinado pelo projeto CNPq 407192/2025-5.

Referências

Acharya, D. B., Kuppan, K., and Divya, B. (2025). Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEE Access*, 13:18912–18936.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462.

Fu, L., Zhang, H., Gao, G., Zhang, M., and Liu, X. (2023). Client selection in federated learning: Principles, challenges, and opportunities. *IEEE Internet of Things Journal*, 10(24):21811–21819.

Jarczewski, R., Cerqueira, E., Bittencourt, L. F., A. F. Loureiro, A., A. Villas, L., and De Souza, A. M. (2026). Participation is power: Effective approach to dynamic federated learning. In *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC '25, New York, NY, USA. Association for Computing Machinery.

Jarczewski, R. O., Cerqueira, E., Bittencourt, L. F., Loureiro, A. A. F., Villas, L. A., and de Souza, A. M. (2024). Let's federate - effective communication strategy for dynamic client participation. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 361–368.

Jee Cho, Y., Wang, J., and Joshi, G. (2022). Towards understanding biased client selection in federated learning. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings*

of The 25th International Conference on Artificial Intelligence and Statistics, volume 151 of *Proceedings of Machine Learning Research*, pages 10351–10375. PMLR.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

Lai, F., Zhu, X., Madhyastha, H. V., and Chowdhury, M. (2021). Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pages 19–35.

Maciel, F., de Souza, A. M., Bittencourt, L. F., Villas, L. A., and Braun, T. (2024). Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Nguyen, D. T., Johnson, T. T., and Leach, K. (2024). Fisc: Federated domain generalization via interpolative style transfer and contrastive learning. *CoRR*, abs/2410.22622.

Nishio, T. and Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE.

Piao, H., Wu, Y., Wu, D., and Wei, Y. (2024). Federated continual learning via prompt-based dual knowledge transfer. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Shah, S. M. and Lau, V. K. N. (2023). Model compression for communication efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):5937–5951.

Souza, A., Bittencourt, L., Cerqueira, E., Loureiro, A., and Villas, L. (2023). Dispositivos, eu escolho vocês: Seleção de clientes adaptativa para comunicação eficiente em aprendizado federado. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

Talasso, G., de Souza, A., Guidoni, D., Cerqueira, E., and Villas, L. (2025). Fine-tuning eficiente de modelos de linguagem para detectar anomalias em logs privados usando aprendizado federado. In *Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 126–139, Porto Alegre, RS, Brasil. SBC.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., and Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xu, J. and Wang, H. (2021). Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE Transactions on Wireless Communications*, 20(2):1188–1200.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.