Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

Daniel Enos Cavalcanti Rodrigues de Macedo

Um Algoritmo de Aprendizagem Federada para Aplicações com Mobilidade e Não-Estacionárias

Campina Grande - PB

2024

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Um Algoritmo de Aprendizagem Federada para Aplicações com Mobilidade e Não-Estacionárias

Daniel Enos Cavalcanti Rodrigues de Macedo

Tese de Doutorado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação
Linha de Pesquisa: Engenharia da Computação

Angelo Perkusich, D.Sc.
Danilo Freire de Souza Santos, D.Sc.
(Orientadores)

Campina Grande, Paraíba, Brasil

M141a Macedo, Daniel Enos Cavalcanti Rodrigues de.
Um algoritmo de aprendizagem federada para aplicações com mobilidade e não-estacionárias / Daniel Enos Cavalcanti Rodrigues de Macedo. – Campina Grande, 2024.
163 f. : il. color.

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr. Angelo Perkusich, Prof. Dr. Danilo Freire de Souza Santos".
Referências.

1. Aprendizagem Federada. 2. Processamento da Informação. 3. Não-estacionariedade. 4. Aprendizagem de Máquina. 5. Engenharia da Computação. 6. Inteligência Artificial. 7. Mobilidade. 8. Desvio de Aprendizagem. I. Perkusich, Angelo. II. Santos, Danilo Freire de Souza. III. Título.

CDU 621.391:004.8(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

Um Algoritmo de Aprendizagem Federada para Aplicações com Mobilidade e Não-Estacionárias

DANIEL ENOS CAVALCANTI RODRIGUES DE MACEDO

TESE APROVADA EM 30/08/2024

ANGELO PERKUSICH, Dr, UFCG
Orientador(a)

DANILO FREIRE DE SOUZA SANTOS, Dr., UFCG
Orientador(a)

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Examinador(a)

ALEXANDRE JEAN RENÉ SERRES, D.Sc., UFCG
Examinador(a)

ADRIAO DUARTE DORIA NETO, Dr., UFRN
Examinador(a)

ALISSON VASCONCELOS DE BRITO, D.Sc., UFPB
Examinador(a)

JAIDILSON JÓ DA SILVA, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

1. ATA DA DEFESA PARA CONCESSÃO DO GRAU DE DOUTOR EM CIÊNCIAS, NO DOMÍNIO DA ENGENHARIA ELÉTRICA, REALIZADA EM 30 DE AGOSTO DE 2024

(Nº 381)

CANDIDATO(A): **DANIEL ENOS CAVALCANTI RODRIGUES DE MACEDO.** COMISSÃO EXAMINADORA: ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG - Presidente da Comissão e Examinador Interno, ANGELO PERKUSICH, D.Sc., UFCG e DANILO FREIRE DE SOUZA SANTOS, Dsc., UFCG Orientadores, ALEXANDRE JEAN RENÉ SERRES, D.Sc., UFCG Examinador Interno, JAIDILSON JÓ DA SILVA, D.Sc., UFCG Examinador Externo, ADRIÃO DUARTE DÓRIA NETO, Dr., UFRN Examinador Externo, ALISSON VASCONCELOS DE BRITO, D.Sc., UFPB Examinador Externo. TÍTULO DA TESE: Um Algoritmo de Aprendizagem Federada para Aplicações com Mobilidade e Não-Estacionárias. ÁREA DE CONCENTRAÇÃO: Processamento da Informação. HORA DE INÍCIO: **09h00** – LOCAL: **Auditório do Embeeded** e **Sala Virtual, conforme Art. 5º da PORTARIA SEI Nº 01/PRPG/UFCG/GPR, DE 09 DE MAIO DE 2022.** Em sessão pública, após exposição de cerca de 45 minutos, o(a) candidato(a) foi arguido(a) oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua tese, obtendo conceito APROVADO. Face à aprovação, declara o presidente da Comissão, achar-se o examinado, legalmente habilitado a receber o Grau de Doutor em Ciências, no domínio da Engenharia Elétrica, cabendo a Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o(a) mesmo(a) faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, Leandro Ferreira de Lima, e os membros da Comissão Examinadora. Campina Grande, 30 de Agosto de 2024.

LEANDRO FERREIRA DE LIMA

Secretário

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG

Presidente da Comissão e Examinador Interno

ANGELO PERKUSICH, D.Sc., UFCG

Orientador

DANILO FREIRE DE SOUZA SANTOS, Dsc., UFCG

Orientador

ALEXANDRE JEAN RENÉ SERRES, D.Sc., UFCG
Examinador Interno

JAIDILSON JÓ DA SILVA, D.Sc., UFCG
Examinador Externo

ADRIÃO DUARTE DÓRIA NETO, Dr., UFRN

Examinador Externo

ALISSON VASCONCELOS DE BRITO, D.Sc., UFPB

Examinador Externo

DANIEL ENOS CAVALCANTI RODRIGUES DE MACEDO

Candidato

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Tese de Doutorado da candidato **DANIEL ENOS CAVALCANTI RODRIGUES DE MACEDO**, assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da tese e tomaram conhecimento do teor deste documento.

Documento assinado eletronicamente por **LEANDRO FERREIRA DE LIMA**, **SECRETÁRIO (A)**, em 02/09/2024, às 14:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **ALEXANDRE JEAN RENE SERRES**, **COORDENADOR DE POS-GRADUACAO**, em 02/09/2024, às 16:07, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **DANILO FREIRE DE SOUZA SANTOS**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 02/09/2024, às 19:24, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **JAIDILSON JO DA SILVA**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/09/2024, às 00:18, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **ANGELO PERKUSICH**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/09/2024, às 06:50, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **ANTONIO MARCUS NOGUEIRA LIMA**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/09/2024, às 06:52, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Daniel Enos Cavalcanti Rodrigues de Macedo**, **Usuário Externo**, em 03/09/2024, às 14:16, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Alisson Vasconcelos de Brito**, **Usuário Externo**, em 04/09/2024, às 04:52, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **4759958** e o código CRC **CE277FA7**.

Referência: Processo nº 23096.059177/2024-81 SEI nº 4759958

Resumo

A Aprendizagem Federada é uma técnica de treinamento de modelos de Aprendizagem de Máquina que garante a privacidade dos dados dos dispositivos que os detêm. A privacidade dos dados é assegurada ao atribuir a responsabilidade do treinamento dos modelos aos dispositivos, que usam exclusivamente suas respectivas bases de dados para treinar modelos locais. Essa técnica tem ganhado atenção da ciência e da indústria em contextos onde as aplicações são cada vez mais rigorosas em que termos de requisitos de segurança e privacidade.

Todavia, ao atribuir o treinamento aos dispositivos, a eficiência do treinamento fica sujeita às características desses dispositivos e do contexto em que estão inseridos. Assim, é fundamental analisar essas características e implementar técnicas para garantir a eficiência do treinamento. Entre essas características, destacam-se a mobilidade dos dispositivos e a não-estacionariedade do sistema. A mobilidade dos dispositivos interfere no treinamento ao impedir sua conclusão, devido a consequente perda de conexão com a rede. Por sua vez, a não-estacionariedade do sistema está diretamente relacionada com o desvio e a consequente depreciação da eficiência de aprendizagem do modelo.

Dessa maneira, apresenta-se o MoSimFeL, um algoritmo de coordenação de Aprendizagem Federada específico para garantir a eficiência do treinamento em cenários com mobilidade e sistemas não-estacionários. Para analisar a eficiência do MoSimFeL, desenvolveu-se um simulador denominado FLSimulator. Esse simulador avalia aplicações de Aprendizagem Federada sob o contexto descrito e é usado para analisar experimentalmente o comportamento do MoSimFeL a partir de uma aplicação de classificação de imagens, comparando o algoritmo proposto com suas variações e com um algoritmo mais tradicional. Por fim, os resultados das simulações mostram que o MoSimFeL é capaz de realizar o treinamento federado mesmo em cenários com intensa mobilidade dos usuários e sob sistemas com não-estacionariedade, enquanto outros algoritmos tradicionais são incapazes de fazê-lo.

Palavras-chave: Aprendizagem Federada, Não-estacionariedade, Mobilidade, Desvio de Aprendizagem.

Abstract

Federated Learning is a training technique for Machine Learning models that guarantees the privacy of the data of the devices that hold it. Data privacy is ensured by assigning responsibility for model training to devices that exclusively use their respective databases to train local models. This technique has gained attention from science and industry in contexts where applications are increasingly stringent in terms of security and privacy requirements.

However, when assigning training to devices, training efficiency is subject to the characteristics of these devices and the context in which they are inserted. Therefore, analyzing these characteristics and implementing techniques is essential to ensure training efficiency. Among these characteristics, the devices' mobility and the system's non-stationarity stand out. The mobility of devices interferes with training to prevent completion due to the consequent loss of connection to the network. In turn, the non-stationarity of the system is directly related to the deviation and the consequent depreciation of the model's learning efficiency.

MoSimFeL, a specific Federated Learning coordination algorithm, is introduced as a solution to ensure training efficiency in scenarios with mobility and non-stationary systems. To validate its effectiveness, a simulator called FLSimulator was developed. This simulator evaluates Federated Learning applications in the described context and is used to experimentally analyze the behavior of MoSimFeL in an image classification application. The results of these simulations demonstrate that MoSimFeL is capable of performing federated training even in scenarios with intense user mobility and under non-stationarity systems, a feat that traditional algorithms struggle to achieve.

Keywords: Federated Learning, Non-stationarity, Mobility, Learning depreciation.

Agradecimentos

À Deus por tudo.

Aos meus pais, Ejosivan Macedo e Terezinha Macedo, por minha vida e zelo em minha educação. Desde cedo ensinaram-me o valor e o poder do conhecimento.

À minha esposa Raquel Macedo, por ser minha companheira, amiga e conselheira, que esteve comigo durante todo o processo de pesquisa e abdicou de tanto para me apoiar. Sua compreensão em meus frequentes momentos de ausência para dedicação ao doutorado me inspira a ser um companheiro melhor.

Aos meus orientadores, Prof. Dr. Angelo Perkusich e Prof. Dr. Danilo Santos, pelas discussões e direcionamentos. Enfatizo que vocês me inspiram como profissionais dedicados.

Aos professores, Dr. Antonio Marcus (UFCG), Dr. Jaidilson Silva (UFCG), Dr. Alisson Vasconcelos (UFPB) e Dr. Adrião Durte (UFRN) pelas críticas e sugestões.

Ao Prof. Dr. Dalton Valadares e aos colegas de doutorado Romulo Omena, Marcus Marinho, Victor Emanuel e Daniel Hindemburg, pelo suporte e troca de conhecimentos nos corredores do EMBEDDED e em nossas reuniões de acompanhamento semanal.

Ao EMBEDDED (Laboratório de Sistemas Embarcados e Computação Pervasiva), que ofereceu estrutura física e uma rede de apoio com excelentes profissionais para a execução desta pesquisa.

Aos alunos e docentes do LIT (Laboratório de Inovação Tecnológica) do Instituto Federal do Rio Grande do Norte, em especial ao Prof. Dr. Francisco Júnior, pelo compartilhamento de experiências e motivação.

À COPELE-UFCG, pelo apoio administrativo, em especial ao servidor Leandro Lima.

Agradeço a todos.

Conteúdo

Lista de Símbolos, Abreviaturas e Acrônimos

5G – *Quinta Geração de Redes Móveis*

AM – *Aprendizagem de Máquina*

AF – *Aprendizagem Federada*

SUMO – *Simulation of Urban MObility*

IoT – *Internet of Things*

QoS – *Quality of Service*

CKA – *Central Kernel Aligment*

FCKA – *Federated Central Kernel Aligment*

MSU – *Micro State Unit*

EAP – *Edge Access Points*

RSU – *Road State Units*

HAF – *Hierárquica de Aprendizagem Federada*

IID – *Independentes e Identicamente Distribuídas*

AFMT – *Aprendizagem Federada Multi-tarefa*

AP – *Aprendizagem Personalizada*

CNN – *Convolutional neural network*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Aprendizagem de Máquina (AM) é uma subárea da Inteligência Artificial (IA) que simula o comportamento de aprendizagem da inteligência humana com algoritmos computacionais [33, 42]. Aplicações baseadas em AM estão se tornando cada vez mais comum no dia a dia, sendo utilizadas em diversas áreas de pesquisa, como visão computacional, tomada de decisão, processamento de linguagem natural, computação gráfica e controle inteligente [2, 5, 6, 9, 22, 23, 25, 57].

Um grande desafio no desenvolvimento dessas aplicações é o aprendizado desses modelos, com base em treinamentos supervisionados, não supervisionados ou por aprendizagem baseada em reforço [78]. No treinamento supervisionado, um conjunto de dados rotulados é usado no treinamento do modelo. No treinamento não supervisionado os dados usados no treinamento não são rotulados. Ou seja, o modelo define os rótulos para as amostras do conjunto de dados durante o treinamento. Por fim, na aprendizagem por reforço o modelo aprende à medida que busca maximizar suas recompensas quando realiza decisões corretas [78].

Independente da estratégia de aprendizagem, em aplicação de AM, um aprendizado eficiente é fundamental para garantir o funcionamento adequado da aplicação e, por vezes, é necessário que o algoritmo de AM tenha um base de dados e experiências para aprender. Sobre a associação dos adjetivos “adequado” e “eficiente” à aplicação, destaca-se que, nesta tese, esses termos referem-se ao funcionamento definido em seu desenvolvimento, seja sob uma perspectiva técnica ou de negócio, considerando seus requisitos. Por exemplo, esses termos podem estar associados a uma acurácia mínima do modelo calculada a partir de uma

base de dados de teste ou a um conjunto mínimo de clientes satisfeitos com o uso da aplicação. Portanto, o uso desses adjetivos é amplo e sua definição precisa é atribuída à aplicação e ao seu contexto. Esses adjetivos também são usados para qualificar o treinamento de um modelo e, da mesma forma, são usados de forma ampla, com sua definição específica dependendo do interesse da aplicação. Por exemplo, no caso de treinamentos, a aplicação pode definir que um treinamento é eficiente caso ele alcance uma acurácia mínima com um número máximo de ciclos de treinamento.

Nesse sentido, a aprendizagem desses algoritmos de AM é uma fase importante para o desenvolvimento de aplicações baseadas em AM. Assim, para propiciar uma aprendizagem eficiente[1], diversas aplicações de AM necessitam dos dados dos usuários ou de dispositivos tanto para viabilizar o treinamento dos seus modelos quanto para avaliar o sucesso da aplicação em atender aos seus requisitos. Dessa maneira, é importante que os dados representem com precisão o contexto que a aplicação se deparará durante seu funcionamento. Assim, o uso de dados reais durante o desenvolvimento da aplicação, ou seja, dados com similaridade aos dados que a aplicação se deparará durante seu funcionamento, é mais adequado do que dados artificiais ou coletados pela aplicação de forma centralizada.

Todavia a obtenção desses dados pode esbarrar em requisitos de privacidade dos usuários. Entre os desafios da aprendizagem, a privacidade dos dados tem despertado atenção da indústria e de pesquisadores [94, 104]. Esse desafio interfere diretamente na formação da base de dados usada para o treinamento dos modelos de AM, devido à necessidade de compartilhamento e privacidade.

Em relação ao compartilhamento, o tráfego de dados entre dispositivos favorece a interceptação por agentes maliciosos. Porém, mesmo que uma estrutura de segurança garanta a confiabilidade do compartilhamento, ainda é preciso avaliar a disposição dos usuários em fornecer seus dados para aplicações de AM [104].

É compreensível que o usuário se interesse em uma aplicação eficiente, ou seja, que atendam aos requisitos necessários para o funcionamento adequado aplicação. A priori, caberia ao usuário disponibilizar dados referentes ao uso da aplicação, como investimento

[1] O termo “aprendizagem eficiente” refere-se à aprendizagem de um modelo que alcance as métricas de avaliação do treinamento dimensionadas no desenvolvimento da aplicação e estipuladas para atender a requisitos de perspectiva técnica ou de negócio.

para garantir um treinamento eficiente do modelo que será usado por ele próprio. Porém, por muitas vezes, os dados usados para o treinamento são informações pessoais ou valiosas, justificando a recusa do usuário no compartilhamento.

Essa recusa é intensificada em aplicações que são executadas em um contexto de mobilidade em que os canais de comunicação entre os dispositivos são principalmente sem fio, como GSM e Wi-Fi, favorecendo a interceptação de mensagens por agentes maliciosos [58, 72, 122].

Para exemplificar a necessidade de privacidade de dados, é possível citar aplicações na área da saúde. Nessa área, os dispositivos móveis, como celulares inteligentes e dispositivos vestíveis, tornaram-se onipresentes e possuem sensores capazes de armazenar um grande volume de dados pessoais para monitoramento da saúde dos usuários [95]. A implementação de aplicações baseadas em AM com esses dispositivos permite a estimação do gasto energético, identificação de sinais de fotopletismografia e previsão da qualidade do sono de um usuário [82, 122]. Para garantir o treinamento adequado de algumas dessas aplicações, é necessário colaboração na disponibilização desses dados. Todavia esse compartilhamento possui restrições legais e éticas que dificultam sua disponibilização, além de serem consideradas sensivelmente íntimos, desestimulando a proatividade do usuário em disponibilizar seus dados [3, 4, 26].

Por sua vez, em algumas aplicações de veículos autônomos, os algoritmos de AM podem realizar tarefas comuns de direção autônoma, como a predição de rotas com melhor trafegabilidade, o reconhecimento de sinais de trânsito e a identificação de faixas de tráfego. Para isso, a cooperação entre os veículos e o aproveitamento das comunicações veiculares pode auxiliar na melhoria de desempenho dos algoritmos de AM [10, 117]. A percepção colaborativa permite que veículos autônomos troquem dados de sensores entre si para obter resultados melhores dos modelos de máquina, como a classificação de objetos em imagens, sendo um meio eficaz para melhorar a precisão da percepção de veículos autônomos [113].

Todavia, o compartilhamento de dados entre veículos sem qualquer proteção pode provocar o vazamento de informações privadas [112]. Assim, preocupações relacionadas com a privacidade dos dados são um desafio importante para ser resolvido, nesse tipo de aplicação. O desenvolvimento de estratégias para garantir a privacidade dos dados compartilhados e evitar ataques cibernéticos tornam-se fundamentais para viabilizar o uso das técnicas de AM

para os veículos [10, 113, 114].

A necessidade de privacidade de dados também está presente em aplicações de Internet das Coisas (no inglês, *Internet of Things* - IoT) [38]. Nessa área, a AM é empregada para resolver problemas na infraestrutura das redes IoT, como engenharia de tráfego, gerenciamento de rede, segurança, classificação de tráfego da Internet e alocação de recursos computacionais na rede [23]. Além disso, a AM é amplamente utilizada na camada de aplicação [23], como no transporte inteligente em cidades, incluindo otimização de rotas e estacionamento [121], monitoramento de saúde de humanos [32] e aplicações na indústria [8, 53].

Além da natureza privativa dos dados dos usuários e dispositivos, a privacidade também pode ser justificada pelas regras de negócio e monetização da aplicação. Por exemplo, dados de aplicações que rastreiam a posição geográfica de um consumidor através do sinal de GPS de celulares e outros dispositivos vestíveis de IoT. Através dessas coordenadas geográfica, é possível identificar comportamentos de consumo de um indivíduo. Com essas informações, aplicações de AM podem estimular o consumo ao sugerir eficientemente propagandas. Por exemplo, ao identificar que um usuário permanece durante um longo período em uma loja de calçados, pode-se inferir o interesse em comprar esse produto. Assim, ao passar próximo a lojas de calçados, o cliente poderia receber notificações em seu smartphone com promoções estimulando assim o desfecho de uma eventual compra.

O perfil consumista também pode ser inferido de forma mais ampla, observando o padrão de frequência de um usuário em determinados locais e inferindo seu padrão de consumo. Por exemplo, usuários que passam longos períodos em academias de ginástica podem se interessar mais por alimentos saudáveis ou moda esportiva do que usuários que nunca frequentam esse tipo de ambiente. Quando um usuário disponibiliza seus dados para o treinamento de um modelo, ele contribui para o mesmo modelo que é usado para impulsionar o consumo de diversos outros usuários. Dessa maneira, o usuário entende que suas informações são monetizadas e, ao invés de voluntariamente fornecer os dados, exige uma recompensa por esse compartilhamento.

Nesse contexto conflitante entre a privacidade de dados e a necessidade de fomentar o aprendizado de algoritmos, surge a técnica Aprendizagem Federada (AF) para viabilizar o treinamento de algoritmos de AM respeitando o requisito de privacidade. Nessa técnica, os

dados usados para o treinamento do modelo não são compartilhados entre dispositivos e a posse desses dados é restringida exclusivamente aos respectivos dispositivos que geraram ou coletaram os dados.

A AF tem ganhado cada vez mais espaço em aplicações de AM com requisitos de privacidade, desde a divulgação do primeiro algoritmo de coordenação do treinamento federado denominado de FedAvg [80]. Todavia, o treinamento federado apresenta desafios que podem inviabilizar essa técnica [84]. Nessa tese, aborda-se dois deles: o abandono de clientes mediante a mobilidade [34, 74, 75] e o dinamismo de cenários não-estacionários [20, 29, 30, 36].

Para contribuir na solução destes desafios, nesta tese, propõe-se um novo algoritmo de AF, denominado de MoSimFeL (sigla derivado do inglês, *MObility Similarity FEderated Learning*). O MoSimFeL é um algoritmo de coordenação de AF que visa garantir o treinamento eficiente de um modelo de AF em contextos de mobilidade e não-estacionários, a partir da seleção enviesada de clientes. Para o MoSimFeL a eficiência de modelos é conseguida a partir de retreinamentos, priorizando lições mais novas e mantendo o aprendizado coerente com as mudanças de contexto da aplicação.

Para tal, para o MoSimFeL dois aspectos são avaliados: o histórico de treinamento dos clientes e a disponibilidade do cliente em iniciar e concluir o treinamento local. A avaliação do histórico inclui a participação dos clientes no treinamento de outros modelos, considerando a similaridade entre as instâncias de modelos anteriores e o modelo a ser retreinado. Por sua vez, na avaliação de disponibilidade analisa-se o empenho de recursos computacionais para o cliente iniciar e concluir o treinamento local federado, considerando a sua mobilidade.

Em relação à mobilidade, para o MoSimFeL rotas dos clientes são avaliadas antecipadamente com a perspectiva de inferir a capacidade do cliente concluir o treinamento antes que sua posição provoque a mudança do servidor central. Nesse contexto, a mobilidade é conjuntamente avaliada com o tempo estimado para o cliente executar o treinamento, considerando sua disponibilidade de recursos computacionais.

Considerando um ambiente não-estocástico, o algoritmo avalia indiretamente o quanto a base de dados de um cliente candidato pode influenciar no treinamento sem violar o requisito de privacidade dos dados. Para tal, é analisada a contribuição do cliente em seu último treinamento local e a similaridade entre o modelo global resultante desse treinamento e o

modelo a ser atualizado. De forma didática, nessa análise investiga-se o questionamento: *Se um cliente contribuiu de determinada maneira em um treinamento anterior, como ele contribuirá em um novo treinamento se for selecionado?*.

Para analisar a similaridade, é utilizada uma variação da métrica de similaridade *Central Kernel Aligment* (CKA)para comparar os modelos treinados de AM [86]. A variação é denominada de *Federated Central Kernel Aligment* (FCKA). O FCKA possui a mesma base matemática do CKA, todavia sua implementação segue uma linha distribuída e todos os servidores centrais participam do cálculo da métrica. Assim, os modelos de AM de cada um dos servidores centrais não são compartilhados entre si, restringindo o tráfego dessas informações em diferentes redes. Apesar do FCKA trazer modificações sutis em relação ao CKA, essas mudanças permitem que o MoSimFeL reforce o princípio de privacidade, visto que a interceptação dos gradientes de treinamento dos clientes e dos modelos globais podem ser usados para inferir a base de dados dos clientes e trazer riscos ao princípio de privacidade por agentes maliciosos [71].

Para avaliar a influência da base de dados de um cliente no retreinamento de um modelo, é analisado o gradiente resultante do treinamento local que contribuiu na agregação do modelo. O gradiente do treinamento local é compartilhado entre o cliente e o servidor central e o uso dessa informação não infringi o princípio de privacidade.

A fim de avaliar a eficiência do MoSimFeL, experimentos foram executados a partir da simulação de uma aplicação de AM baseada em uma Rede Neural Convolucional (no inglês, *Convolutional neural network*, CNN). Nesses experimentos, o MoSimFeL é comparado ao FedAvg. Analisando os resultados das simulações, é possível concluir que o MoSimFeL alcança uma acurácia mínima definida por uma aplicação com menos ciclos de treinamento. Com isso, consegue-se otimizar a alocação de recursos computacionais e o consumo de banda no canal de comunicação. Além disso, as análises dos resultados das simulações possibilitam concluir que o MoSimFeL atende as demandas de novos clientes conectados a um servidor central a partir do retreinamento do modelo.

As simulações foram executadas no FLSimulator[2] [77], um software de simulação de algoritmos de coordenação de AF capaz de simular cenários com mobilidade e não estacionários. Durante as simulações, o FLSimulator calcula diferentes métricas de eficiência

[2]Disponível em: https://github.com/enosmacedo/SimulatorFL - Acesso em: 23 de Julho de 2024.

desses algoritmos, como o número de abandonos de treinamento, a avaliação da acurácia pelos clientes e o número de ciclos de treinamento executados. O FLSimulator foi concebido como um arcabouço de códigos computacionais bem estruturados, permitindo a extensão para diferentes algoritmos de coordenação e cenários de simulação. Além disso, é importante destacar que o FLSimulator é uma contribuição desta tese.

Nas próximas seções, as concepções introdutórias desta tese são apresentadas de forma sistematizada. Na Seção 1.1, descreve-se detalhadamente o escopo do problema abordado. Na Seção 1.2, discute-se a justificativa para o desenvolvimento desta pesquisa, respaldando sua importância no contexto de AM. Na Seção 1.4, relata-se o caminho percorrido nesta pesquisa, destacando o percusso que levou ao estudo do problema da Seção 1.1 e à concepção do desenvolvimento do MoSimFeL. Na Seção 1.5, enfatiza-se as contribuições resultantes da pesquisa desenvolvida. Finalizando a introdução, na Seção 1.6, sumariza-se a organização do restante da tese, auxiliando sua leitura.

1.1 Problemática

Aplicações de AM podem exigir sigilo dos dados de treinamento e a restrição ao acesso dos dados somente pelos seus proprietários legítimos. Para isso, a técnica de AF atribui aos clientes, proprietários dos dados, a tarefa de treinamento dos modelos a partir da base de dados individual de cada cliente.

Na AF há dois papéis importantes para que o treinamento do modelo de máquina ocorra: os clientes e o servidor central. Na aplicação, os clientes (que são os usuários) assumem a responsabilidade de treinar os modelos de AM localmente, alocando recursos computacionais para realizar esta tarefa. O resultado do treinamento de cada cliente é compartilhado com um servidor central. O servidor central agrega todos os resultados na definição de um modelo global que é compartilhado com todos os clientes da rede. Portanto, o modelo global é o resultado do treinamento colaborativo. Além de agregar o modelo global, o servidor central é responsável por coordenar o treinamento, selecionando os clientes que participarão da etapa de treinamento local.

Apesar das contribuições para viabilizar requisitos de privacidade no treinamento de modelos de AM, o uso dessa técnica enfrenta desafios como o problema de abandono de clientes

devido à mobilidade e o dinamismo de cenários não estacionários.

Em relação ao abandono de clientes, esse problema ocorre quando um cliente selecionado para o treinamento local não compartilha o resultado do seu treinamento com o servidor. Nesse contexto, a mobilidade pode acarretar o abandono quando propicia a quebra de comunicação entre o cliente e o servidor central, impossibilitando o compartilhamento do resultado do treinamento.

Durante a movimentação de um cliente, é possível que ele migre de rede a fim de garantir melhor QoS (*Quality of Service*, em inglês) ou para atender a requisitos de negócio da aplicação e, com isso, se conecte a um novo servidor central. Nesse caso, o cliente muda de domínio de servidor central [3].

O abandono de um único cliente durante um ciclo de treinamento não é tão danoso para a aprendizagem, mas é prejudicial para esse cliente, que empenhou esforços e recursos computacionais para o treinamento local, mas não contribui na definição do modelo global. Todavia, o aumento do número de abandonos impacta diretamente no tempo de convergência do aprendizado e na acurácia do modelo. Nesse caso, problemas como *underfitting* e *overfitting* [27], amplamente conhecidos e abordados nas técnicas tradicionais de treinamento de AM, podem acontecer na AF. Partindo para um caso extremo, a disponibilidade de clientes na rede pode ser tão pequena que inviabiliza a aprendizagem ao excluir amostras de um grupo de classes do conjunto de dados do treinamento.

Outro desafio da AF é o treinamento de modelos em ambientes não estacionários. Nesses ambientes, as propriedades probabilísticas dos dados mudam com o passar do tempo, repercutindo na função a ser aprendida, de forma que as lições aprendidas anteriormente deixam de ser importantes, precisando ser substituídas por novos aprendizados ou coexistir com novas lições [29, 30, 79, 97].

Em relação a não-estacionariedade, no contexto da AF, o cliente é um provável autor pelo dinamismo da aplicação, pois comumente essas aplicações possuem bases de dados heterogêneas, não-IID (não Independentes e Identicamente Distribuídos) e dinâmicas à medida que

[3] Nesta tese, os termos *domínio de servidor central* e *domínio da rede* referem-se ao conjunto de dispositivos conectados, incluindo um servidor central e clientes, que compartilham um modelo global, no qual o servidor central controla seu treinamento. Por sua vez, os termos *domínio da aplicação* e *domínio do problema* referem-se ao contexto em que a aplicação está submetida, incluindo o conjunto de dados e o arranjo de usuários e servidores.

são mantidas e atualizadas pelos clientes, que por sua vez também são heterogêneos entre si. Nesse sentido, é importante enfatizar que a não-estacionariedade não é uma consequência exclusiva do treinamento federado, mas também do contexto da aplicação [29, 36, 79].

O contexto não-estacionário pode ser causado, entre outros motivos, por efeitos da sazonalidade, mudanças no *hardware* que afetem a aplicação, como falhas ou envelhecimento, e preferências do usuário, que acontecem a partir de mudanças graduais ou de tendência, mudanças abruptas, contextos ocultos ou contextos recorrentes [30, 79]. Por exemplo, um sistema de identificação de faixas de tráfego por processamento de imagens para aplicação de tráfego autônomo de veículos inteligentes sofre fortes influências de mudanças climáticas, variação do tipo de solo e mudança de horário (devido a influência na iluminação do ambiente). Essas mudanças repercutem diretamente nas características de imagens adquiridas pelas câmeras dos veículos, influenciando nos resultados de algoritmos de reconhecimento das faixas. Nesse mesmo contexto, um pequeno inseto pode obstruir repentinamente e parcialmente a lente da câmera, inutilizando parte das imagens captadas.

Apesar do desafio de aprendizado em ambientes não-estacionários estar presente em diferentes aplicações de AM, é ainda mais comum em aplicações de AF, visto que os dados empenhados na aprendizagem do modelo são oriundos dos clientes que por vezes apresentam diferentes perfis para aquisição de dados. Ou seja, possuem sensores, capacidade de armazenamento e periodicidade diferentes entre outras características.

Nesse sentido, além da heterogeneidade de *hardware* interferir diretamente na formação da base de dados do cliente, o comportamento desse cliente também é importante. Por exemplo, a primeira aplicação de AF foi destinada aos usuários de smartphones e tinha o objetivo de aprender novas palavras e frases a partir da digitação no teclado desses aparelhos para auxiliar na digitação. Evidentemente, o volume de textos digitados por um usuário varia com base no perfil do aparelho e do usuário, impactando diretamente na coleta de dados e aumentando a heterogeneidade da aplicação [80].

A não-estacionariedade e a mobilidade também estão diretamente relacionadas. Em muitas aplicações sem fio e sob o contexto de mobilidade, a distribuição de dados sensoriais não é apenas heterogênea, mas também não-estacionária devido à natureza aleatória de conexões devido à mobilidade dos clientes [126]. Ou seja, há uma relação direta de causa e efeito, respectivamente, entre a mobilidade e a não-estacionariedade, de forma que as aplicações

com mobilidade são propensas a serem ambientes não-estacionários. Assim, o cliente é um provável autor pelo dinamismo da aplicação, pois comumente essas aplicações possuem bases de dados heterogêneas, não-IID e dinâmicas à medida que são mantidas e atualizadas pelos clientes, que por sua vez também são heterogêneos entre si. Nesse sentido, é importante enfatizar que a não-estacionariedade não é uma consequência exclusiva do treinamento federado, mas também do contexto da aplicação [29, 36, 79].

Outra característica que interfere diretamente no treinamento é a ocorrência de multidomínios, que são, por muitas vezes, consequência de um contexto não-estacionário, com aspectos de mobilidade e com distribuição de dados não-IID. É importante garantir que a aplicação seja eficiente independentemente do domínio a que esteja submetida e, portanto, o modelo de AM deve manter os requisitos de eficiência definidos pela aplicação.

Com base na descrição dos desafios dessa seção, nesta pesquisa, busca-se responder às seguintes questões de pesquisa:

1. O quanto a mobilidade dos clientes interfere no treinamento do modelo de Aprendizagem Federada?

2. Como a não-estacionariedade interfere na eficiência da AF?

3. Como garantir um treinamento federado eficiente em contexto de mobilidade, não-estacionariedade e multidomínios?

Para orientar esta pesquisa, ajudando a responder adequadamente a essas questões de pesquisa, o problema pode ser definido de duas maneiras:

- Problema de Negócio - viabilizar aplicações de AM com requisitos de privacidade dos dados a partir do treinamento eficiente de modelos de AM em cenários de mobilidade, com multidomínios e não-estacionários.

- Problema Técnico - Investigar, desenvolver e aplicar técnicas de seleção de clientes na AF que considerem aspectos de mobilidade e a base de dados dos clientes em cenários de mobilidade e não-estacionários, amenizando problemas com o enviesamento do modelo de AM, retardação do treinamento e esquecimento catastrófico, garantindo a eficiência do treinamento de modelos de AM.

1.2 Justificativa

Conforme mencionado no início do Capítulo 1, uma das etapas mais importantes do treinamento de um AM é a formação do conjunto de dados usados no treinamento. Em algumas aplicações essa base de dados é formada com o auxílio do usuário que coletam ou geram os dados. Todavia, a disponibilização desses dados pode ser um empecilho para os usuários que julgam serem valiosos ou que possuam requisitos de segurança que não permitam o compartilhamento de dados na rede. Para lidar com o desafio de divulgação de dados, a técnica de AF atribui o treinamento do modelo aos usuários da aplicação, mantendo a posse dos dados exclusivamente sob usuários que os coletaram e impedindo o compartilhamento. À medida que o usuário assume a responsabilidade do treinamento, desde a preparação da base dos dados, o modelo fica suscetível a características deste usuário. Entre essas características, a mobilidade e a não-estacionariedade podem interferir no treinamento do modelo.

Nesses ambientes, um modelo treinado com uma base de dados construída a partir de uma perspectiva estacionária fatalmente sofrerá depreciação, ficando obsoleto e diminuindo sua eficiência na aplicação [30]. Portanto, dado o crescente número de aplicações concebidas sob processos não-estacionários de geração de dados, é fundamental aprimorar os algoritmos de treinamento de AF com procedimentos de detecção e adaptação a não-estacionariedade a fim de dar robustez aos algoritmos e mitigar a desatualização e a incapacidade de generalização de modelos AM mediante as mudanças dinâmicas do contexto ao longo do tempo [36].

Dessa maneira, para garantir a evolução de aplicações de AM com requisitos de privacidade dos dados dos usuários em cenários com mobilidade, é fundamental avaliar e propor estratégias de treinamento federado robustos para enfrentar problemas relacionados à mobilidade dos usuários e capazes de amenizar os custos computacionais relacionados ao treinamento. Caso contrário, o modelo da aplicação corre o risco de sofrer depreciação tornando-se ineficaz de atender aos requisitos da aplicação. A necessidade da atualização do aprendizado de um modelo é ainda mais evidente em aplicações cujos requisitos são criticamente rígidos mediante a natureza da aplicação, de modo que, o não cumprimento dos requisitos podem causar prejuízos financeiros ou até mesmo risco à vida humana [6, 115].

É evidente a necessidade de considerar a mobilidade e a não-estacionaridade em algoritmos de coordenação de AF garantir simultaneamente a AM e a privacidade de dados.

Todavia, apesar dos avanços na literatura com o desenvolvimento de novas técnicas de AF, há uma lacuna, até onde foi pesquisado, para suprir essa necessidade. De forma que, os atuais algoritmos identificados na literatura não abordam esses desafios ou são superficiais ao tratá-los.

Assim, esta tese atua no desenvolvimento de uma solução que atenda a três importantes desafios simultaneamente: privacidade de dados, mobilidade e ambientes não-estacionários. Considerando o atual desenvolvimento de aplicações que tenham essas três características, esta pesquisa impulsiona importantes contribuições para o desenvolvimento científico e industrial, à medida que colabora com a viabilidade de implementação da AM em aplicações com esse perfil.

1.3 Objetivo Geral

O objetivo geral é viabilizar aplicações de AM com requisitos de privacidade de dados e contrárias ao compartilhamento de dados de usuários, considerando a mobilidade e a não-estacionariedade.

1.3.1 Objetivos Específicos

A fim de alcançar o objetivo geral são definidos os seguintes objetivos específicos:

1. investigar algoritmos de AF;
2. enumerar soluções e desafios presentes na literatura acadêmica sob a AF;
3. propor um algoritmo de AF robusto ao abandono de treinamento e em ambientes não-estacionários;
4. investigar técnicas de AM para classificação de imagens em tempo real;
5. desenvolver um simulador para avaliar a eficiência de novos algoritmos de coordenação de AF sob o cenário de mobilidade e não-estacionariedade;
6. comparar os algoritmos propostos com algoritmos já conhecidos, tais como o FedAvg, a fim de avaliar a eficiência desses algoritmos em relação aos desafios definidos.

1.4 Metodologia

Nesta seção, aborda-se a metodologia utilizada no desenvolvimento deste trabalho. Assim, narra-se em ordem cronológica as fases de pesquisa que foram vivenciadas para a construção da problemática e concepção das soluções apresentadas nesta tese.

Ao todo, o desenvolvimento é dividido em 4 fases, conforme representado pelo esquema da Figura 1.1 que sumariza os destaques de cada fase. Nas subseções seguintes, cada uma das fases será detalhadamente descrita, especificando as motivações e desafios encontrados, assim como, os resultados da pesquisa oriundos da investigação, incluindo contribuições para solução dos desafios apresentados.

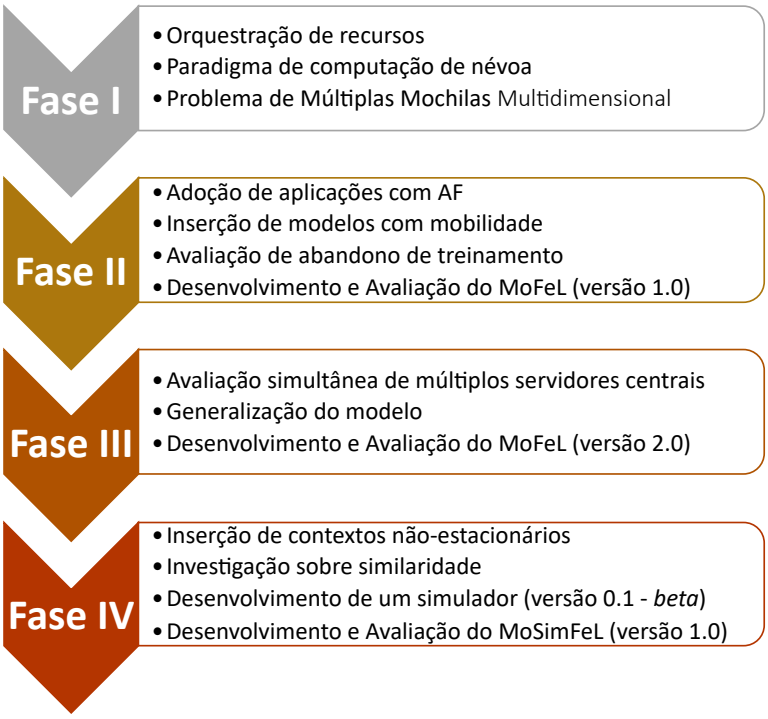


Figura 1.1: Esquema metodologia

1.4.1 Fase I

O desenvolvimento deste trabalho foi iniciado com uma investigação da literatura sobre a disponibilização de recursos computacionais para dispositivos móveis com requisitos críticos de latência. Durante a investigação, observou-se a necessidade de avaliar a orquestração de recursos na emergente tecnologia 5G [35, 62]. O direcionamento para essa tecnologia

foi justificado pelo seu processo de inovação e implementação de infraestrutura, soluções e aplicações no Brasil.

O 5G se enquadra na arquitetura inicialmente investigada, sendo facilmente implementável em contextos dos paradigmas de computação de borda [54] e computação de névoa [11, 83]. Nesse sentido, percebeu-se a necessidade de descrever matematicamente um modelo para formalizar um sistema de orquestração de recursos, considerando restrições pré-estabelecidas.

Assim, na primeira publicação científica resultante do processo de desenvolvimento dessa tese, foi proposto um modelo para orquestração de recursos em Macedo et al. [76]. Neste trabalho, um método de orquestração de recursos para computação de borda, que atribui requisições a servidores de borda, tendo em vista as demandas das requisições e o custo de alocação. Esse método é baseado em um modelo matemático de otimização *min-max* com restrições que consideram o tempo de duração da alocação de recursos, o tempo de latência requerido pela requisição e o custo da alocação e tem o objetivo de otimizar o custo de alocação atendendo aos requisitos da requisição.

A modelagem foi inspirada no clássico problema de otimização da medida, sendo classificada especificamente como um Problema de Múltiplas Mochilas Multidimensional [18, 52]. Com a solução do problema de otimização, o método decidia onde uma requisição deveria ser alocada (nuvem ou borda), explorando o dinamismo do paradigma de névoa e orquestrando recursos através de servidores de névoa que gerenciam a comunicação entre usuários e os servidores de borda e da nuvem, conforme é exemplificado no cenário da Figura 1.2.

A fim de exemplificar uma aplicação proposta em Macedo et al. [76], na Figura 1.2 são apresentadas esquematicamente pequenas redes com diferentes aplicações, como uma rede de semáforos inteligentes que organizam o tráfego de uma cidade e uma rede de dispositivos conectados em uma indústria para controle de produção. Os recursos computacionais estão disponíveis na borda e na nuvem e os usuários podem solicitar recursos para requisições computacionais para execução de algoritmos, armazenamento de dados e outros.

A função objetivo definida para o problema de otimização define a alocação de recursos computacionais com base no custo e é definida pela Equação 1.1, onde x_{mnt} é uma variável binária, onde 1 implica que uma requisição n deve ser alocada no servidor m. De forma

Fonte: Adaptado de Macedo et al. [76]

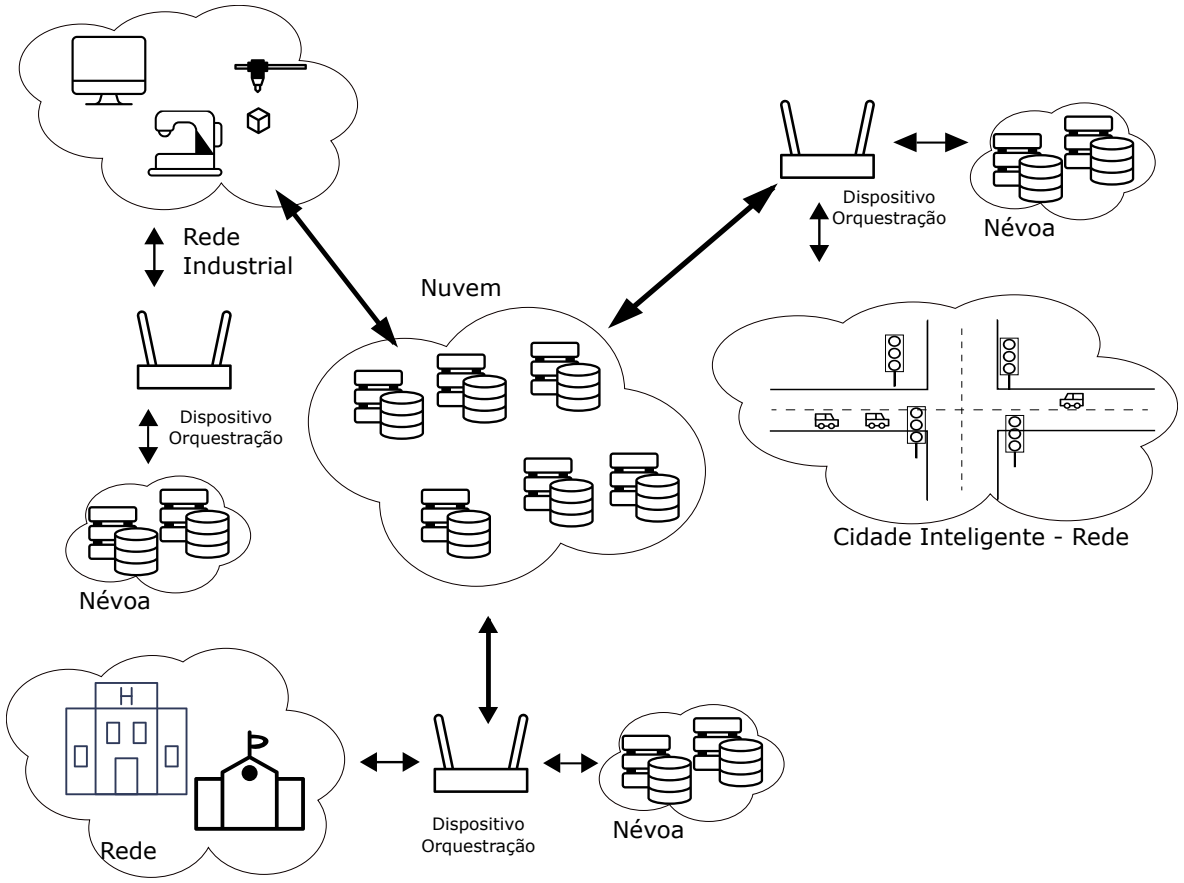


Figura 1.2: Exemplo de topologia com diferentes aplicações e cenários.

análoga, o valor 0 implica que essa alocação não deve acontecer. Por sua vez, $d_{n,r}$ é a exigência do recurso r por uma requisição n, tal que $d_{n,r} \in \Re$ e $p_{m,r}$ é a disponibilidade de recurso r por um servidor m, tal que $p_{m,r} \in \Re$. Por fim, T representa um intervalo discreto de tempo em que o método que a orquestração avalia para sua decisão.

$$\textit{min} \left(\sum_{t=0}^{T} \sum_{i=0}^{M} \sum_{j=0}^{N} x_{ijt} \cdot \left(\sum_{r=0}^{|R|} p_{i,r} - d_{j,r} \right) \right), \tag{1.1}$$

onde $x_{ijt} \in \{0, 1\}$, R é um conjunto de recursos que podem ser requisitados por um dispositivos, tal que r ($r \in R$), M é o conjunto de servidores e N é o conjunto de requisições a serem orquestradas.

Uma avaliação por simulação mostrou que o método proposto era mais eficiente do que estratégias gulosas. Todavia, foi enfatizado que havia um custo computacional necessário

para resolver o problema de otimização, recomendando-se uma análise mais aprofundada para melhorias computacionais sobre o método. Além disso, não foi avaliado o efeito da mobilidade no cenário, considerando, por exemplo, a perda de comunicação entre um usuário e um servidor de borda escolhido para atender às requisições do usuário. Caso ocorra essa perda, o serviço disponibilizado deveria ser transferido para uma outra borda para garantir a latência exigida, mas isso não foi analisado.

Após a avaliação da orquestração de recursos na borda, concluiu-se que as restrições para alocação dos recursos dependem dos requisitos da aplicação. Dessa forma, a extensão desse trabalho para implementação de uma técnica eficiente dependerá da aplicação.

Conclusões da Fase Metodológica

Nesse contexto, iniciou-se uma nova fase do desenvolvimento desta pesquisa (Fase II) cujo objetivo inicial foi compreender como as aplicações interferiram e se relacionavam com as restrições e definições de recursos. Diferentes aplicações foram analisadas a partir de uma revisão da literatura para elencar os requisitos da aplicação.

1.4.2 Fase II

A Fase II desta pesquisa tinha como objetivo analisar diferentes aplicações para compreender quais recursos computacionais precisariam ser orquestrados na borda. A perspectiva era que essa compreensão resultaria em um algoritmo de orquestração otimizado para cada aplicação, considerando seus requisitos. Durante esta fase, a execução da pesquisa cruzou com a recente técnica de AF.

A técnica de AF era uma tecnologia emergente e recente, podendo ser considerada assim ainda hoje. Essa aplicação destacou-se, entre outros, pela definição de recurso computacional. Ao definir que os usuários da aplicação (os clientes) assumem o treinamento local do modelo, a maior necessidade de recursos computacionais do treinamento federado se concentra no usuário em vez de concentrar-se em dispositivos centralizados na borda. Ou seja, de certa forma, isso inverte o problema até então estudado e previsto no paradigma de computação de borda. Antes, na computação de borda tradicional, os usuários solicitam e atribuem demandas aos servidores na borda. Agora, na AF, são os servidores que solicitam

aos usuários a execução de demandas computacionais.

Nesse contexto, a fim de garantir a eficiência de aplicações de AF em contexto de mobilidade, iniciou-se ainda na Fase II uma investigação para analisar quais as consequências que a mobilidade poderia surtir nessas aplicações. Considerando até onde a investigação alcançou, concluiu-se que não havia na literatura avaliações precisas e suficientes que analisassem a mobilidade no contexto de AF.

Assim, identificou-se a necessidade de implementação de um simulador para avaliação de AF em cenários com mobilidade. Para a simulação foi utilizada uma estrutura de grafo (V, E) para controlar a migração [4] de conexão de clientes , onde V é um conjunto de vértices que representa o domínio de um servidor e E é um conjunto de arestas (simbolizado por (u, v), onde $u \in V$ e $v \in V$) que representa a possibilidade de migração de um cliente do domínio u para o v e vice-versa, a depender do requisitos da aplicação ou do cenário simulado. A migração de um cliente entre os domínios (u, v) ocorre considerando a existência da aresta e a ocorrência de um evento estatístico com base em uma distribuição estocástica, tal como uma distribuição Guassiana ou Normal.

Com base nos resultados de simulações, observou-se que à medida que se aumentava o índice de mobilidade, o algoritmo tradicional de AF perdia a capacidade de concluir o treinamento, devido ao abandono de clientes causado pela migração de conexão de servidor central oriunda da mobilidade de clientes. Deste modo, foi proposto um novo algoritmo, denominado MoFeL para avaliação de AF que considerasse aspectos de mobilidade de clientes para melhorar a aprendizagem dos algoritmos, sendo divulgado em Macedo et al. [74].

A principal diferença entre o FedAvg e MoFeL está na etapa de seleção de clientes. Enquanto para o FedAvg realiza-se a seleção aleatoriamente, no cado do MoFeL as rotas dos clientes informadas por eles mesmos antecipadamente e voluntariamente são consideradas, permitindo que um servidor central avalie, em um intervalo de tempo, a melhor configuração de disponibilidade de clientes para agendar o início de um treinamento com a perspectiva que no instante agendado, o arranjo de clientes conectados ao domínio desse servidor se concretize. Além das rotas, os clientes informam a disponibilidade de recursos computacionais para treinamento dos modelos para que o servidor busque uma solução mais eficiente

[4] Nesta tese, utilizamos o termo migração para nos referirmos à alteração da conexão de um cliente entre servidores centrais [74, 75]

para o treinamento, buscando a conclusão do treinamento em um menor intervalo de tempo.

Além disso, para o MoFeL, foi definida uma estratégia de inicialização de treinamento do modelo, enquanto que, no caso do FedAvg, não há uma definição de como isso deve ser feito. Nas Figuras 1.3 e 1.4 mostra-se como os ciclos de treinamento são executados no caso do MoFeL considerando as atribuições dos clientes e do servidor central, respectivamente.

Na Figura 1.3, ao ingressar em um novo domínio (passo 1), um cliente executa o carregamento do modelo global (passo 2) de um servidor central quando se conecta a ele. Ou seja, cada servidor central mantém uma instância própria do modelo e controla exclusivamente o treinamento dele. Durante o uso do modelo, o cliente avalia constantemente a sua eficiência (passos 3 e 4) e notifica ao servidor central caso considere que o modelo não atende mais as suas perspectivas com base nos requisitos da aplicação (passo 5).

Como apresentado na Figura 1.4, o MoFeL inicia (passo 1) o processo de AF com o servidor central solicitando a avaliação dos clientes de seu domínio sobre seu modelo (passo 2). Considerando as perspectivas de todos os clientes, o servidor central define se o modelo permanece adequado ou não para os requisitos da aplicação (passo 3). Caso o servidor central defina que o modelo não é adequado, ele entra em estado de alerta (passo 4) para iniciar um novo treinamento.

No passo 5, o servidor central solicita a todos os clientes do seu domínio que enviem informações sobre a sua rota e disponibilidade de recursos. Com essas informações, um problema de otimização *min-max* (Equação 1.2) é solucionado para selecionar os clientes (passo 7).

O objetivo da Equação 1.2 é maximizar o ganho na inferência do cliente e definir, como consequência, o instante para iniciar a execução do treinamento local.

$$f = max\left(\sum_{n}^{N} x_n \cdot w_n \cdot (1 - p_n)\right), \tag{1.2}$$

onde x_n é uma variável e representa a decisão se um cliente n deve ser selecionado para o treinamento local, tal que $x_n \in \{0, 1\}$, $x_n = 1$ representa uma resposta positiva para a seleção e $x_n = 0$ representa o inverso, N é o conjunto de clientes, tal que $n \in N$. w_n e p_n representam a capacidade de n de contribuir com o treinamento, na perspectiva do próprio cliente e do servidor central, respectivamente, logo $w_n \in \mathbb{R}$ e $p_n \in \mathbb{R}$.

No restante dos passos do fluxograma da Figura 1.4, o MoFeL é similar ao FedAvg após a fase de seleção do cliente. Portanto, os passos $[8-12]$ destacados na Figura 1.4 são similares ao FedAvg.

Fonte: Adaptado de Macedo et al. [74]

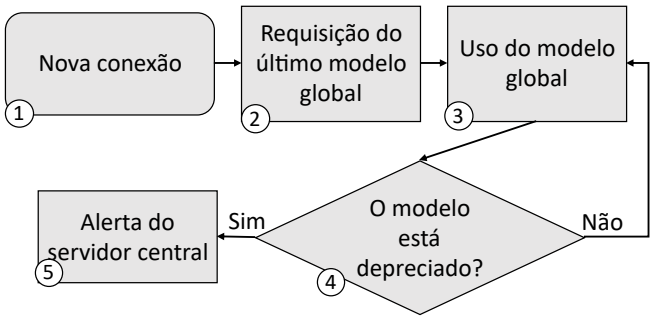


Figura 1.3: Ciclo de vida do cliente da AF do MoFeL.

Fonte: Adaptado de Macedo et al. [74]

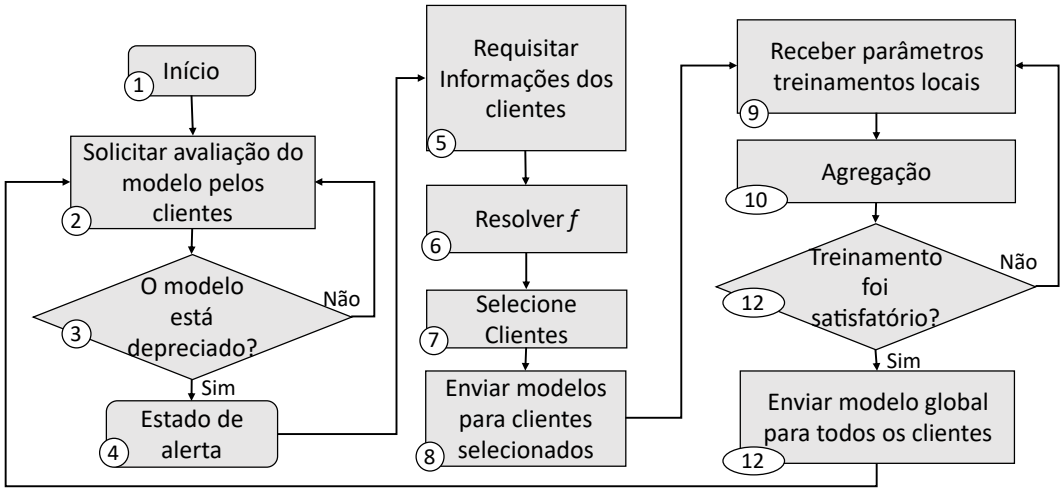


Figura 1.4: Ciclo de vida do servidor central da AF do MoFeL.

Conclusões da Fase Metodológica

Assim, o MoFeL herda do método da Fase I a estratégia de agendar o treinamento para que, ao iniciar o treinamento local, o domínio do servidor central detenha de um arranjo de clientes mais adequado. O MoFeL também incorpora uma estratégia para orquestração de

recursos com base na mobilidade já que considera a rota dos clientes e, na AF, os clientes podem ser considerados os recursos da aplicação.

Concluindo a Fase II, observou-se que os clientes com menor mobilidade tinham maiores chances de serem selecionados, assim como os clientes que possuem mais recursos, pois estes tinham maior perspectiva de concluir o treinamento na de intervalo de tempo analisado. Dessa maneira, alguns clientes poderiam ser solicitados repetidamente para o treinamento, sobrecarregando-os por disponibilizar recursos computacionais repetidamente. Além disso, surgia o risco de enviesar o modelo à medida que as bases de dados dos mesmos clientes eram repetidas no treinamento.

Dessa forma, optou-se em avaliar uma solução panorâmica que pudesse controlar de forma mais eficiente a alocação de clientes para o treinamento local em diversas redes simultaneamente. Esse marco define o início da Fase III.

1.4.3 Fase III

Na Fase III, a compreensão da Equação 1.2 foi expandida a fim de analisar o problema de seleção de clientes com múltiplos servidores centrais, resultando em uma nova versão do MoFeL. Para tal, um novo dispositivo ao cenário, denominado de estação central, foi definido. Esse dispositivo é responsável por avaliar panoramicamente os servidores centrais que vislumbram executar um novo treinamento. Assim, ele seleciona os clientes simultaneamente para todos os servidores centrais e auxilia no agendamento do treinamento de cada servidor.

Assim, no novo problema, a Equação 1.2 foi substituída pela Equação 1.3.

$$f = min\left(\sum_{s \in S} F_s\right), \tag{1.3}$$

onde F_s é a função de erro do modelo global do servidor central s e o objetivo é minimizar a soma desses erros, considerando que cada servidor central mantém um modelo único.

Observe que o valor de F_s só pode ser definido após a conclusão do treinamento, uma vez que o treinamento de um modelo de AM é um evento não determinístico. Logo, na Equação 1.3, F_s precisa ser aproximada de forma preditiva para viabilizar a solução do problema

de otimização. Em Macedo et al. [75], definiu-se arbitrariamente como requisito de aplicação que F_s é diretamente proporcional ao número de clientes que participam do treinamento. Todavia, no trabalho foi destacado a importância de avaliar F_s a partir de outras perspectivas.

Devido ao custo computacional para a solução da Equação 1.3, os autores desse trabalho destacaram a importância de equilibrar a modelagem do problema considerando a descrição de restrições, o custo para solução do problema e a manutenção de um espaço de solução viável para encontrar uma resposta que atenda aos requisitos da aplicação [75].

A versão do MoFeL da Fase III também foi avaliada através de simulações que seguiram um fluxo similar a Fase II. Os resultados da simulação mostraram que o MoFeL consegue executar o treinamento mesmo em cenários com intensa mobilidade pelos clientes, diferente do FedAvg.

Conclusões da Fase Metodológica

A avaliação das simulações da Fase III indicou a importância de considerar a mobilidade dos clientes na etapa de seleção. Todavia, percebeu-se que a avaliação da mobilidade era uma inferência da capacidade do cliente concluir o treinamento e, para melhorar ainda mais o treinamento, seria necessário avaliar indiretamente a base de dados dos clientes. Assim, a pesquisa da Fase III elucidou novos caminhos que direcionaram para novas pesquisas. Os apontamentos dessa fase levaram ao seguinte questionamento: *Como antecipar a contribuição de um cliente selecionado em um ciclo de treinamento federado?*

1.4.4 Fase IV

Nos trabalhos desenvolvidos, até então, assumiu-se cenários em que os clientes formavam a sua base de dados a partir de um contexto estacionário. Nesse sentido, percebeu-se que essa concessão limitava a avaliação da aplicação em contextos reais.

Na Fase IV, o desenvolvimento dessa tese expandiu a avaliação de aplicações AF para cenários não-estacionários, incluindo os requisitos anteriores como mobilidade, múltiplos-domínios e disponibilidade de recursos. Assim, iniciou-se o estudo relacionado aos conceitos de similaridade entre modelos de AM e em paralelo investigou-se como esse conceito estava sendo utilizado na AF. Com o estudo levantado, concluiu-se que o conceito já estava

sendo aplicado no contexto de AF. Porém a interseção dos três temas: mobilidade, não-estacionariedade e múltiplos-domínios.

Para avaliar as soluções propostas para o desafio dos novos cenários, desenvolveu-se um simulador mais robusto que interage com as ferramentas já consolidadas na literatura. Uma versão preliminar foi registrada em Macedo et al. [77]. Diferente do simulador desenvolvido nas Fases II e III, o simulador da Fase IV consegue simular cenários de mobilidade mais próximos a cenários reais.

Com o novo simulador e, consequentemente, simulações de mobilidade mais próximas a cenários reais, verificou-se que a solução de um problema de otimização *min-max* similares as Equações 1.2 e 1.3 não eram computacionalmente viáveis e não não traziam melhoras significativas para a coordenação do treinamento federado nesses cenários. Isso porque os simuladores de mobilidade conseguem emular o erro de previsão de rotas, de forma que a proposta de encontrar uma solução ótima ficou inviável, dada a imprecisão das rotas dos clientes. Todavia, a avaliação de rota ainda deve ser mantida com a perspectiva de encontrar clientes aptos para o treinamento, mesmo com incertezas.

Conclusões da Fase Metodológica

Devido à cronologia de execução das fases e ao aprendizado que as fases consecutivas herdam das fases anteriores, as soluções e desafios propostos nesta tese estão mais próximos da execução da Fase IV. Todavia, enfatiza-se que esta tese é o resultado do desenvolvimento de todas as fases descritas anteriormente. Assim, os resultados e conclusões desta fase serão descritos detalhadamente ao longo desta tese.

1.5 Contribuições

Até onde foi possível observar na literatura, o MoSimFeL é o primeiro algoritmo de AF a avaliar a similaridade entre modelos globais e o histórico de participação dos clientes em treinamentos locais na seleção de clientes. Além disso, é o primeiro algoritmo a avaliar a disponibilidade dos cliente em diferentes domínios a partir de sua mobilidade, amenizando o abandono no treinamento por perda de conexão com o servidor central [74]. Em geral, trabalhos anteriores limitaram-se a avaliar o histórico dos clientes na seleção sem considerar

quesitos de similaridade. Esse trabalho também é pioneiro em avaliar simultaneamente o comportamento de algoritmos de coordenação de AF a partir da análise de múltiplos domínios [75].

Neste contexto, as principais contribuições desta tese são resumidas a seguir:

1. propõe-se um algoritmo de AF para treinamento de modelos de AM a partir do histórico de atuação dos clientes em treinamento de outros modelos e da disponibilidade de clientes em diferentes domínios;

2. define-se uma metodologia de execução do CKA para garantir a privacidade dos clientes no treinamento;

3. infere-se a eficácia da participação de um cliente em treinamento de modelo a partir de seu histórico de treinamentos anteriores em outros modelos;

4. apresenta-se um simulador de AF capaz de simular cenários com mobilidade;

5. avalia-se o algoritmo MoSimFeL comparativamente com o FedAvg e outros algoritmos a fim de analisar sua eficiência.

Como evidência dos resultados encontrados na pesquisa oriunda desta tese, enumera-se as publicações:

1. Daniel E. Macedo, Marcus M. Bezerra, Danilo F. S. Santos, and Angelo Perkusich. Orchestrating fog computing resources based on the multi-dimensional multiple knapsacks problem. In Leonard Barolli, editor, *Advanced Information Networking and Applications*, pages 318–329, Cham, 2023. Springer International Publishing. ISBN 978-3-031-28451-9.

2. Daniel Macedo, Danilo Santos, Angelo Perkusich, and Dalton Valadares. A mobility-aware federated learning coordination algorithm. *J. Supercomput.*, 79(17): 19049–19063, may 2023. ISSN 0920-8542. doi: 10.1007/s11227-023-05372-3. URL https://doi.org/10.1007/s11227-023-05372-3.

3. Daniel Macedo, Danilo Santos, Angelo Perkusich, and Dalton C. G. Valadares. Mobility-aware federated learning considering multiple networks. *Sensors*, 23(14),

2023. ISSN 1424-8220. doi: 10.3390/s23146286. URL https://www.mdpi.com/1424-8220/23/14/6286.

4. Daniel E. Macedo, Danilo F. S. Santos, and Angelo Perkusich. Simulatorfl, 2024. URL https://github.com/enosmacedo/SimulatorFL.

1.6 Organização do Texto

O restante desta tese é dividido da seguinte forma:

- No Capítulo 2, apresenta-se a técnica de AF (Seção 2.1) acompanhada de uma discussão sobre os seus principais desafios e soluções presentes na literatura, uma discussão sobre similaridade (Seção 2.2) e uma discussão sobre aprendizagem de máquina em sistemas não-estacionários (Seção 2.3). O propósito desse capítulo é explicar os conceitos técnicos abordados nesta tese. Para isso, elenca-se trabalhos presentes na literatura para auxiliar na explicação e na contextualização desses conceitos.

- No Capítulo 3, discute-se uma revisão da literatura. Com essa revisão, analisa-se aspectos dos problemas levantados nesta tese, como sintetizado na Seção 1.1, considerando a descrição detalhada do problema e soluções já apresentadas na literatura. Além disso, no final desse capítulo, aspectos frágeis dos trabalhos revisados e que precisam ser abordados para solucionar os problemas propostos são elencados. Diferente do Capítulo 2, o foco deste capítulo é a análise da literatura relacionada com o problema desta tese e não na explicação dos conceitos e técnicas usadas nesta tese.

- No Capítulo 4, explica-se detalhadamente o algoritmo MoSimFeL para coordenação da AF a fim de amenizar os problemas de mobilidade dos clientes e desvio de aprendizagem em sistemas não-estacionários.

- No Capítulo 5, apresenta-se um simulador para aplicações de AF em contextos de mobilidade e não-estacionários. O simulador é usado na análise do MoSimFeL.

- No Capítulo 6, apresenta-se simulações, comparando o comportamento do MoSimFeL com outros algoritmos. Além disso, discute-se o resultado das simulações criticamente, avaliando a eficiência da técnica proposta.

- No Capítulo 7, retoma-se as principais conclusões do trabalho. Além disso, elenca-se trabalhos futuros.

Capítulo 2

Aprendizagem Federada, Similaridade e Não-estacionariedade, um Arcabouço Conceitual

Neste capítulo, os principais conceitos e técnicas empregados nesta tese são apresentados de forma detalhada. Nesse sentido, os principais conceitos que norteiam esse trabalho são: privacidade de dados, similaridade e não-estacionariedade. Assim, na Seção 2.1, explica-se o principal algoritmo de AF e os principais desafios do treinamento federado. Na Seção 2.2, apresenta-se o conceito de similaridade e uma revisão da literatura sobre esse conceito. Por fim, na Seção 2.3, apresenta-se o conceito formal de não-estacionariedade. Na Figura 2.1, expõe-se esquematicamente a organização dos conceitos apresentados neste capítulo.

Acrescenta-se que, em cada seção, aborda-se uma revisão da literatura com trabalhos relevantes para essa tese de forma que ao final do capítulo diversos trabalhos abordados fundamentam a proposta do MoSimFeL.

2.1 Visão Crítica Sobre Aprendizagem Federada

O objetivo principal da técnica da AF é viabilizar o treinamento de modelos com a garantia de privacidade e posse dos dados de treinamento. Nessa técnica, os dados de treinamento são coletados separadamente por diferentes clientes e não são compartilhados com nenhum outro dispositivo, garantindo total posse dessas informações.

Figura 2.1: Diagrama esquemático com a organização de conceitos do Capítulo 2.

Para garantir a posse dos dados, o cliente treina localmente o modelo de máquina exclusivamente com seus dados e compartilha o resultado do treinamento com um dispositivo central. Por sua vez, o dispositivo central agrega os resultados dos treinamentos locais de todos os clientes em um modelo global.

O primeiro algoritmo de coordenação da AF foi denominado de *Federated Averaging Algorithm* (FedAvg) e foi implementado em aplicações para sugestão de entrada de texto em teclados de tela sensível ao toque, em dispositivos móveis, melhorando a digitação dos usuários [80].

Mediante o seu pioneirismo, o FedAvg tornou-se um dos principais algoritmos de comparação para os algoritmos da AF que surgiram posteriormente e é comumente usado como algoritmo de referência para descrever a técnica da AF.

Assim, a descrição do FedAvg, através do Algoritmo 1, sistematiza a descrição formal da AF [16, 48, 66, 80, 88, 109].

Como já mencionado, há dois papéis na técnica de AF: o cliente e o servidor central. Em relação ao cliente, a técnica inicia com a coleta de dados. Essa coleta é independente e cada dispositivo tem a posse dos seus dados, não compartilhando com outros dispositivos. Na descrição do FedAvg [80], não há detalhamento de qual a rotina de aquisição desses dados

Algorithm 1 Algoritmo FedAvg

Require: s: Servidor central; C: conjunto de clientes; $|C_N|$: número de clientes selecionados; $|R|$: quantidade ciclos de treinamento;

1: **for** $r \in |R|$ **do**
2: s seleciona $|C_N|$ clientes, tal que $C_N \subseteq C$
3: s envia modelo global M_g atualizado para C_N
4: **for** $c \in C_N$ **do**
5: Pré-processamento de dados
6: Treinamento do novo modelo local M_c a partir de M_g
7: Envio de M_c para s
8: **end for**
9: s atualiza M_g através de agregação
10: s avalia M_g
11: **if** M_g é adequado **then**
12: Finalizar treinamento
13: **end if**
14: **end for**
15: **return** M_g

pelos clientes, incluindo o tamanho de cada base de dados ou a periodicidade de atualização da base de dados e um acordo entre a aplicação e seus clientes deve definir como isso é feito. Para o FedAvg, também não define-se estratégias de pré-processamento da base dedados antes do treinamento, como execução de filtros e complementação da base a partir de dados artificiai [49, 56]. Todavia, a linha 5 representa o pré-processamento dos dados como uma possibilidade para os clientes executarem um pré-processamento antes do treinamento local.

Enquanto os clientes coletam dados, o servidor central se comunica com os clientes verificando a disponibilidade de recursos computacionais e de dados para treinamento. Caso seja adequado, a partir dessas informações, o dispositivo inicia um ciclo de treinamento. Novamente, quando o treinamento deve ser executado também não é especificado em [80] até porque isso está relacionado diretamente com especificidades da aplicação.

A primeira etapa (linha 2) do ciclo é selecionar os clientes para executar o treinamento lo-

cal. Na proposta do FedAvg, a seleção de clientes ocorre aleatoriamente e sem enviesamento algum. Além disso, não é descrito quantos clientes serão selecionados. Após a seleção, os clientes recebem uma cópia dos parâmetros do modelo atualizado da aplicação caso ainda não a tenham (linha 3) e iniciam imediatamente o treinamento local unicamente com seus dados (linha 6).

Ao concluir seu respectivo treinamento, cada cliente envia ao servidor central os parâmetros resultantes (linha 7) e agrega os parâmetros em um modelo único através de uma média ponderada considerando o tamanho da base de dados que cada cliente empenhou em seu treinamento (linha 9). O modelo resultante da agregação é denominado de modelo global e representa a contribuição de todos os clientes a partir dos treinamentos de cada base de dados. Ainda sobre as etapas de agregação e treinamento local, é importante enfatizar que o FedAvg é síncrono e os parâmetros são agregados apenas após receber as contribuições dos clientes selecionados. Ou seja, o modelo global só é definido após todos os clientes selecionados finalizarem o treinamento local.

O modelo global é avaliado pelo servidor central (linha 10) e, a partir da avaliação, define-se a necessidade de mais ciclos de treinamento para melhorar o treinamento do modelo global ou se o modelo já está adequado para a aplicação (linha 12). Novamente, na descrição inicial do FedAvg não foi defido o mérito dessa decisão.

Com a conclusão dos ciclos de treinamento, o servidor central compartilha o modelo global para todos os usuários da aplicação (linha 15).

A implantação eficiente da AF enfrenta diversos desafios que são intensificados ou amenizados a partir dos requisitos da aplicação e o contexto em que será implementado. Nesse sentido, diversas pesquisas focaram no desenvolvimento da AF preservando o princípio da privacidade de dados.

Para organizar a produção literária relacionada a AF, diferentes trabalhos da literatura, já categorizam os desafios e soluções da aprendizagem federada em rótulos como: grande consumo de banda para comunicação, sistema heterogêneo, heterogeneidade estatística, preocupações com privacidade dos dados, enviesamento do treinamento e abandono de clientes [13, 63]. Todavia, a rápida expansão da área, com o crescente número de trabalhos, requer uma nova organização para a compreensão ampla dos desafios.

Nesse contexto e seguindo o diagrama da Figura 2.1, nas próximas subseções, apresenta-

se uma revisão da literatura considerando desafios e contribuições para a AF. A categorização dos artigos abordados nessa seção parametrizou o objetivo das soluções e os desafios tratados em cada artigo e, em cada subseção, aborda-se uma dessas categorias. Obviamente, a solução abordada em um trabalho pode mirar em vários objetivos e soluções simultaneamente, classificando-se em mais de uma categoria. Quando isso acontecer, o trabalho é categorizado a partir da eleição de um foco principal, destacando-se as demais contribuições sem perda de generalidade, visto que a divisão de seções culmina apenas na organização didática dessa tese.

2.1.1 Comunicação

Segundo Li et al. [61], o principal gargalo do treinamento federado é a dificuldade de diminuir o custo de comunicação resultante do compartilhamento de modelos na rede por diversas vezes até alcançar a precisão do modelo. Isso acontece pois há um esforço no gerenciamento dos ciclos de treinamento que resulta em um grande volume de dados compartilhados entre o servidor central e os clientes para a atualização do modelo global e do compartilhamento dos parâmetros do treinamento local. O desafio sobre o consumo de banda de comunicação é ainda mais compreensível em aplicações realísticas em que, comumente, o treinamento da AF persiste por diversos ciclos devido a limitação da base de dados dos clientes, falhas no retreinamento e outros fatores [13, 63]. Por exemplo, Nguyen et al. [85] concluíram que o custo de comunicação influenciava diretamente no consumo de bateria de dispositivos móveis ao executar o treinamento local. Apresentando uma solução, foi proposto pagar o custo de energia através de *beacon* de energia para a recarga desses dispositivos e uma *Deep Q-Network* foi desenvolvida para encontrar uma solução ideal para otimizar o custo com gasto de energia, maximizando o número de transmissões e minimizando os custos de energia e de canal.

De forma objetiva, as contribuições para esses problemas permeiam pela diminuição de ciclos necessários para o treinamento do modelo e a redução do tamanho das mensagens de cada ciclo [63].

A fim de reduzir os custos de comunicação, Bouacida et al. [14] propuseram uma nova técnica denominada de *Adaptive Federated Dropout*, reduzindo o tempo de convergência do treinamento do modelo. A técnica proposta permite que cada cliente treine um submodelo

enquanto ainda fornece atualizações aplicáveis ao modelo global maior no lado do servidor. Os custos de comunicação para o treinamento desses submodelos menores entre os clientes e o servidor são menores do que manter atualizações completas do modelo. Além disso, os clientes precisam empenhar menos recursos computacionais no treinamento local, visto que esses submodelos são menores.

2.1.2 Sistema Heterogêneo

O treinamento local do modelo nos clientes é executado utilizando recursos computacionais (como memória e processador) dos clientes. Todavia os dispositivos na rede são heterogêneos e se diferenciam sobre a disponibilidade de recursos entre si [48, 63]. Além disso, em muitas aplicações, os clientes não dispõem de recursos computacionais específicos e dedicados para esse treinamento e, por isso, a disponibilidade de recursos para atender a essa demanda varia com o passar do tempo.

A restrição na disponibilidade de recursos computacionais pelos clientes acarreta uma fração reduzida de clientes aptos a contribuir com o treinamento federado. Dessa forma, para os algoritmos de coordenação de AF, deve-se levar em consideração, por exemplo, a heterogeneidade dos dispositivos, o treinamento sob baixa quantidade de dispositivos e a perda repentina de dispositivos [63].

Nesse contexto, Zhang et al. [123] propuseram um algoritmo de controle orientado à experiência que adequa a seleção de clientes para cada rodada da AF, através de um modelo de Processo de Decisão Markov, sem conhecimento prévio do ambiente. A seleção de clientes é baseada em aprendizado por reforço que aprende a selecionar um subconjunto de dispositivos em cada rodada de comunicação para minimizar o consumo de energia e o atraso de treinamento, incentivando o aumento do número clientes no aprendizado.

Ainda a fim de amenizar os problemas acarretados pela heterogeneidade dos clientes na rede, em uma rede veicular, Xiao et al. [110] formularam um problema de otimização *min-max* para otimizar a capacidade de computação a bordo, levando em consideração a posição, a velocidade do veículo, a potência de transmissão e a precisão do modelo local. O objetivo da função é atingir o custo mínimo no pior caso da AF.

Liu et al. [67] propuseram um mecanismo da AF assíncrono adaptativo a fim de reduzir o treinamento de clientes retardatários que atrasam a agregação global do modelo. Esse atraso

é causado pela disponibilidade efetiva de recursos computacionais para o treinamento assim como o volume de dados empenhados para o treinamento, visto que o tamanho da base é diretamente relacionado ao tempo de processamento. Para isso, o servidor central agrega apenas uma fração dos resultados gerados pelo treinamento local. Essa fração é equalizada através de aprendizado de reforço profundo para alcançar o valor ideal e, consequentemente, obter tempos de conclusão de treinamento menores.

Para amenizar o atraso, ao invés de uma seleção aleatória, em Jee Cho et al. [47] foi proposto uma seleção tendenciosa para interferir na velocidade de convergência no treinamento. Além de quantificar como o viés afeta a velocidade de convergência, mostrou-se que direcionar a seleção para clientes com maior erro de treinamento local resulta em uma convergência de erros mais rápida. Com objetivo similar, o algoritmo FedProx é uma alteração do FedAvg que permite que cargas de treinamento distintas sejam atribuídas a diferentes clientes, reconhecendo que os clientes são heterogêneos e que a distribuição igual de carga de trabalho provoca atrasos, justificados pela menor disponibilidade de recursos computacionais. [7].

É importante destacar que o atraso de um cliente no treinamento não é consequência exclusiva da heterogeneidade dos dispositivos, mas também pela disponibilização relativa de dados no sistema, que favorece a conclusão do treinamento em clientes em detrimento ao prolongamento de outros, falhas de comunicação, incluindo o congestionamento do canal de comunicação entre o cliente e o servidor central, bem como outros fatores.

Wen et al. [107] também reconheceram o desafio de lidar com clientes que possuem escassos recursos computacionais para o treinamento local e propôs um esquema de abandono federado para poda de modelo aleatório. Na proposta, diversos submodelos são gerados independentemente a partir do modelo global no servidor usando corte de parâmetros a partir de diferentes taxas. Cada submodelo é carregado por um subconjunto de clientes que executam seu treinamento, reduzindo a sobrecarga de comunicação e as cargas de computação dos dispositivos comparativamente ao FedAvg [107].

Por fim, em um contexto de computação de borda móvel, Wen et al. [107] propuseram um particionamento do modelo global, formando diferentes submodelos. Assim, conjuntos distintos de clientes podem treinar um submodelo idêntico ou os clientes podem treinar separadamente diferentes submodelos, integrando os padrões de paralelismo de dados e paralelismo de modelo. O objetivo final é que os clientes móveis, considerando a heterogeneidade

de recursos, podem ser alocados eficientemente no treinamento dos submodelos [107].

2.1.3 Heterogeneidade dos Dados

Além da heterogeneidade intrínseca das características de *hardware* do cliente, o AF também enfrenta a heterogeneidade estatística referente ao contexto em que o cliente está inserido, interferindo diretamente na formação da base de dados do treinamento. A heterogeneidade do sistema e a heterogeneidade na coleta de dados são características da AF que a diferem do treinamento distribuído tradicional de AM. No treinamento distribuído tradicional, os dispositivos que executam o treinamento são conhecidos e a distribuição de dados para treinamento é proporcional a sua disponibilidade de recursos computacionais e pode ser controlada durante o processo [48, 116].

Na AF, em muitas aplicações, a formação da base de dados por clientes ocorre de forma não-IID, em que as bases de dados de cada cliente variam em qualidade, diversidade e quantidade. Esse desafio também é relacionado ao ciclo de vida de dados, principalmente quando é curto, fato perceptível em muitas aplicações modernas, pois perdem sua capacidade de representar o domínio problema da aplicação. Nesses casos, os usuários mantêm uma alta taxa de atualização dos dados aumentando o dinamismo da aplicação e a não-estacionariedade [61, 116].

Todavia, para garantir um modelo de AM com alta taxa de precisão e generalização e um treinamento eficiente, é importante definir estratégias eficientes de coletas de dados que possibilitem um bom treinamento. A disponibilidade de dados de forma irregular para os clientes, devido a imersão em ambientes onde os dados são distribuídos de uma forma não-IID, pode dificultar a coleta de dados à medida que sobrecarrega alguns clientes com um grande volume de dados em detrimento de outros, com menor volume. Outro fator importante, é a qualidade do conjunto de dados de treinamento. É preciso que esse conjunto tenha diversidade de dados, a fim de contribuir para a generalização do modelo e diminuir a quantidade de ciclos de treinamento à medida que aumenta a eficiência de treinamento de cada ciclo e contribui com a diminuição do consumo de banda de comunicação.

Para contribuir com a solução desse desafio, Briggs et al. [15] introduziram uma etapa de agrupamento hierárquico para separar *clusters* de clientes pela similaridade de suas atualizações locais. Uma vez separados, os *clusters* são treinados paralelamente e indepen-

dentemente resultando em modelos especializados para cada *clusters*. Isso permite que o treinamento do modelo convirja em menos ciclos de treinamento, principalmente em cenários não-IID, e um número maior de clientes consiga a acurácia requerida pela aplicação em comparação com AF sem agrupamento.

A estratégia de coleta de dados também pode auxiliar no problema de barreira de sincronização inerente da AF. Esse problema acontece quando opta-se pela agregação síncrona, sendo necessário aguardar que os clientes terminem o treinamento local para um novo ciclo de treinamento do modelo. Essa espera pode ser longa, principalmente em cenários não-IID. Uma solução desse problema é ajustar, de forma adaptativa, o tamanho do conjunto de treinamento de cada cliente com taxas de aprendizagem escalonadas, considerando as características de cada cliente. Assim é possível reduzir o tempo de treinamento, a partir da avaliação dos recursos computacionais disponíveis em cada usuário [73].

Ainda no contexto de amenizar os desafios relacionados a heterogeneidade na distribuição de dados entre os clientes, Zhao et al. [128] propuseram que um conjunto de dados com todas as classes de um problema fossem compartilhados para todos os clientes a fim de amenizar a heterogeneidade na distribuição de dados. Isso é possível para aplicações cujo princípio de privacidade de dados não é tão rígido a ponto de permitir o compartilhamento de um conjunto pequeno de amostras.

Com o mesmo objetivo, soluções tradicionais como a criação de dados artificiais, a partir de dados de vários dispositivos, estão sendo aplicadas na AF, possibilitando o aumento do conjunto de dados de treinamento de um cliente e favorecendo a eficiência do treinamento local. Para isso, foi proposto uma estratégia em que os clientes fabricam artificialmente amostras e permitem o compartilhamento especificamente dessas amostras entre os dispositivos, ou seja, sem compartilhar os dados originais. Com o compartilhamento, a diferença entre as bases de dados de cada cliente diminui, mitigando os efeitos oriundos de distribuições não-IID [124]. Novamente, é possível observar neste trabalho a possibilidade, dentro de aprendizagem federada, relativizar o requisito de privacidade de dados a depender da aplicação. Nesse caso, foi apontado a possibilidade de compartilhamento de dados artificiais.

Por sua vez, Zhang et al. [127] atacaram o desafio da distribuição não-IID também propondo um algoritmo de seleção de clientes aumentando a frequência de seleção dos clientes com menores graus de dados não-IID. No algoritmo proposto, também foi assumido que

parte dos dados do sistema podem ser compartilhados entres os clientes.

2.1.4 Privacidade

O desenvolvimento da técnica da AF foi incitado pela necessidade de garantir a privacidade de dados coletados pelos usuários da rede e que formam a base de dados para a produção do modelo [45, 80]. Nesse sentido, a técnica é eficaz desde que os clientes e servidores da AF não sejam maliciosos [65].

Em relação à segurança, apesar da AF garantir a posse de dados evitando a transmissão das bases de dados de cada cliente pela rede, os requisitos de segurança ainda são uma preocupação. Segundo Brik et al. [16], o compartilhamento de alguns modelos locais permite revelar informações privadas. Nesse sentido, Bonawitz et al. [12] propuseram um algoritmo para criptografar os modelos locais sem a necessidade do servidor central descriptografá-los para executar a agregação. Todavia, ainda é possível adquirir informações sobre a participação de um cliente a partir do modelo global [16].

Hatamizadeh et al. [41] mostrou que alguns ataques para inversão de redes neurais profundas a partir de gradientes não são praticáveis quando o treinamento local usa estatísticas de Normalização de Lote. Além disso, o autor apresentou novas métricas para mensurar vazamentos de dados em AF.

2.1.5 Enviesamento do Modelo

A definição de quantos e quais clientes serão selecionados para o treinamento local é uma das etapas mais críticas de um algoritmo da AF. A escolha repetida dos mesmos clientes para diferentes ciclos da AF pode provocar o enviesamento do modelo global, com perda de acurácia especificamente para o subconjunto de clientes que não participam do treinamento local. Devido à característica não-IID, a distribuição de dados pelos clientes não é uniforme e a exclusão de um cliente em todas os ciclos da AF também acarreta a exclusão de seus dados na definição do modelo global [13].

Uma solução similar é denominada de FedProx [64]. O FedProx é semelhante ao FedAvg, porém modifica a função de erro incluindo parâmetros que podem ser adaptados para restringir o quanto as atualizações locais podem afetar os parâmetros do modelo global. Com

isso é possível ajustar os parâmetros à medida que o erro do treinamento local aumenta, afetando menos o modelo global e garantindo a convergência do modelo [7, 64, 65].

Por sua vez, LadaBoost [44]. é uma técnica adaptativa de reforço em que os clientes comparam entre si o erro dos seus modelos locais e os modelos com maiores perdas de entropia cruzadas são retreinados antes da agregação global, alcançando uma convergência do modelo global mais rápida [44, 65].

Wang et al. [106] propuseram, para modelos de aprendizado de máquina que são treinados usando abordagens baseadas em gradiente descendente, um algoritmo de controle que adapta dinamicamente a frequência da agregação global. Essa decisão é balizada para minimizar a função de perda, considerando os recursos, visto que o treinamento local consome recursos computacionais dos clientes e cada agregação global consome recursos do servidor central.

2.1.6 Abandono de Clientes

O Abandono de Cliente no contexto da AF é a impossibilidade de um cliente selecionado pelo servidor central contribuir para a aprendizagem enviando o resultado do seu treinamento local para a central. Essa impossibilidade pode ser ocasionada pela decisão arbitrária de um cliente em não cumprir com acordo assumido com a central ou por um evento adverso [61, 65, 107]. São exemplos de eventos adversos:

- instabilidade na disponibilidade de recursos computacionais necessários para o treinamento;
- queda do canal de comunicação entre o servidor e o cliente;
- saída do domínio da rede pelo cliente.

Esse último evento está diretamente relacionado com a mobilidade de usuários, principalmente em redes sem fio, visto que a movimentação do cliente pode provocar a perda de conexão com o servidor central. Quando isso acontece, o cliente não consegue mais enviar o resultado do seu treinamento local para o servidor central.

O abandono de um único cliente durante um ciclo de treinamento não é tão danoso para a aprendizagem, mas é prejudicial para esse cliente. Quando um cliente assume o treina-

mento local, ele disponibiliza recursos computacionais para viabilizar essa tarefa, podendo comprometer a execução de outras tarefas do próprio cliente. Logo, quando o treinamento local é executado e não é possível enviar o resultado para a central, a alocação dos recursos computacionais foi inútil.

Nesse contexto, quanto mais escassos são os recursos de um cliente, mais perceptível é o efeito desse prejuízo, como é o caso de aplicações em IoT [81] em que, geralmente, sistemas embarcados e com pouco recursos computacionais assumem o papel de cliente no AF, ou de aplicações móveis em que os recursos energéticos são limitados e há um gasto de energia para viabilizar o treinamento.

Do lado do servidor central, quando apenas um cliente abandona o processo, o modelo global não é tão prejudicado, pois um ciclo de treinamento é formado por diversos outros clientes que conseguem contribuir com o treinamento local. Ou seja, o prejuízo do abandono é diluído pela fidelidade na conclusão de outros treinamentos.

Porém, quando o número de abandonos começa a crescer, a aprendizagem do modelo global pode ser diretamente impactada, repercutindo diretamente no tempo de convergência e na acurácia do modelo. Nesse caso, os mesmos problemas, amplamente conhecidos e abordados nas técnicas tradicionais de treinamento de AM podem acontecer na AF, como o sob ajuste e sobre ajuste (respectivamente, do inglês, *underfitting* e *overfitting*) [27], visto que uma parte significativa da base de dados que culmina na aprendizagem do modelo é comprometida junto com o abandono.

Partindo para um caso extremo, a quantidade de abandonos pode ser tão alta que não restariam clientes suficientes para uma aprendizagem adequada ao diminuir significativamente a base de dados empenhada no treinamento, chegando até a excluir todas as amostras de um classificador.

Nesse sentido, uma proposta óbvia para mitigar o abandono é escolher mais clientes do que o necessário, ou seja, uma margem segura, garantindo a conclusão de um número mínimo de treinamentos locais mesmo ocorrendo o abandono por parte dos clientes. Considerando a causa de mobilidade, nessa proposta, mesmo que uma parte dos clientes percam a conexão com o servidor central, a aprendizagem ainda seria satisfatória em detrimento dos clientes selecionados que mantivessem a conexão. Bonawitz et al. [13] propuseram que o servidor central selecionasse sempre uma quantidade de clientes a mais do que o conside-

rado ideal (em alguns de cenários, foi proposta uma seleção de 130% do número alvo de clientes).

A seleção de clientes adicionais, além de compensar a queda do dispositivo, é uma estratégia para eliminar clientes retardatários. Nesse caso, não existe um abandono do cliente, mas uma decisão premeditada do servidor central em eliminar um cliente durante o ciclo de treinamento [13, 80].

Todavia, é importante atentar que a seleção de uma quantidade exagerada de clientes acarreta a diminuição da taxa de convergência de treinamento. Esse problema é similar às técnicas de treinamento de modelo de máquina quando o conjunto de dados está significativamente grande [13]. Além disso, uma quantidade exagerada de clientes participando de um mesmo ciclo aumenta potencialmente o risco de congestionamento do *uplink* e do custo de comunicação [108] e culmina em uma desnecessária alocação de recursos computacionais em um mesmo ciclo concomitante a falta de recurso nos demais ciclos. Isso acontece, pois os clientes não querem se sobrecarregar dedicando-se ao treinamento e se negam a contribuir nos demais ciclos [60].

Outra solução para evitar o abandono de clientes da rede é definir um índice de confiabilidade e atribuir um valor para cada cliente. Os dispositivos que historicamente apresentam o comportamento de permanecer no mesmo domínio, ou seja, conectados ao mesmo servidor central, durante um ciclo da AF, recebem um índice maior e são considerados mais confiáveis do que outros com o comportamento inverso. Com esses índices é possível que o servidor central mescle na seleção de clientes com diferentes índices para garantir o avanço do treinamento em um ciclo [108]. Essa estratégia é facilmente estendida para outras características do cliente com o propósito de avaliar indiretamente a qualidade da base de dados de um cliente e a disponibilidade de recursos computacionais em executar o treinamento sem provocar atrasos no ciclo. Dessa maneira, a seleção enviesada visa aumentar a eficiência do treinamento em diversos aspectos.

2.1.7 Aplicações

Na área da saúde, foi desenvolvido um modelo para prever hospitalizações por eventos cardíacos através de classificador de máquina de vetor de suporte e técnicas de AF [17]. Por sua vez, Vaid et al. [102] usaram uma técnica de aprendizado de máquina para prever a mor-

talidade em pacientes hospitalizados considerando dados clínicos em prontuários de vários hospitais. Dessa forma, cada hospital utilizava seus dados para treinar localmente modelos para contribuir com o treinamento de um modelo global com maior precisão dos que os modelos locais separadamente.

Em aplicações veiculares, o cache de conteúdo na extremidade das redes veiculares é uma tecnologia promissora para satisfazer as demandas crescentes de aplicativos veiculares com computação intensiva e sensíveis à latência para transporte inteligente. No entanto, para implementar essa solução de forma eficiente é preciso considerar a mobilidade dos veículos conectados a um servidor de borda que torna a popularidade do conteúdo na cache variável e difícil de prever. Além disso, o conteúdo em cache é facilmente desatualizado, pois cada veículo conectado permanece na área de um servidor de borda por um curto período. Para isso, Yu et al. [120] propuseram um esquema de cache proativo de borda com reconhecimento de mobilidade baseado no aprendizado federado. O esquema permite que vários veículos aprendam de forma colaborativa um modelo global para prever a popularidade do conteúdo com os dados de treinamento distribuídos em veículos.

Por sua vez, veículos aéreos não tripulados (VANT) são cada vez mais comuns em diversas aplicações militares e civis [93, 125]. Todavia alguns veículos possuem limitações quanto a sua capacidade de transporte e reserva energética, que dificulta voos longos e tarefas árduas, como o transporte de cargas grandes. Uma solução para essa limitação é usar diversos VANTs de forma colaborativa para a solução de uma tarefa.

É comum nesse contexto, que os dispositivos se conectem a uma base fixa no solo através de um canal sem fio e essa base coordene os VANTs para alcançar o objetivo da aplicação, com disponibilização de recursos computacionais ou executando algoritmos a partir de informações individuais dos dispositivos.

Esse cenário remonta a arquiteturas de computação de borda que são facilmente empregadas em aplicações com AF. Por exemplo, Zhang and Hanzo [125] implementaram uma aplicação para classificação de imagens coletadas a partir das câmeras de bordo dos VANTs. Utilizando a AF, a aplicação treinou modelos de máquina e reduziu o custo de comunicação entre os VANTs e a central.

Nesse sentido, segundo Tang et al. [100], VANTs podem ajustar o empenho de recursos computacionais da CPU, alocando recursos computacionais na borda, para diminuir o uso da

bateria. Dessa forma, é possível evitar o abandono no treinamento de aprendizado federado. Para isso, uma estratégia baseada em gradiente de política determinístico profundo otimiza a alocação de recursos computacionais e a largura de banda sem fio em ambientes que variam no tempo.

2.2 Similaridade

Diferentes trabalhos da literatura propuseram contribuições para AF com emprego de similaridade. O algoritmo FedSim foi um dos primeiros trabalhos a utilizar o conceito de similaridade no contexto do AF e apresentou uma métrica de heterogeneidade dos dados que pode ser inferida através de similaridade entre os gradientes dos clientes. Além disso, demonstrou que essa métrica é diretamente relacionada ao desempenho e convergência do AF [91]. No FedSim, é definido *clusters* de clientes, através do algoritmo *Kmeans++* [91], com o objetivo de cada *cluster* agregar modelos intermediários que posteriormente também serão agregados para a formação do modelo global. A avaliação de similaridade é feita através dos gradientes resultantes dos treinamentos locais a cada rodada e isso pode aumentar o custo computacional devido aos cálculos de similaridade entre os pares. Para amenizar esse problema, foi proposto reduzir a dimensionalidade dos gradientes através da Análise de Componentes Principais antes do *Kmeans++*. Dessa maneira o treinamento alcança uma melhor convergência ao reduzir a variância entre os modelos agregados quando comparado ao FedAvg e FedProx.

Briggs et al. [15] avaliar o resultado de cada treinamento local, ou seja, o gradiente com o objetivo de agregar gradualmente os modelos locais por agrupamento. Para isso, o gradiente vetorizado define um *cluster* contendo unicamente sua amostra. Ou seja, considerando que N é o número de vetores, há inicialmente N *clusters*. Durante a análise de similaridade, é executado o agrupamento de *clusters* por etapas. Em cada etapa, a distância entre pares de *clusters* é calculada para avaliar sua similaridade, de forma que, os dois *clusters* mais semelhantes são agrupados em único *cluster*, resultando em um *cluster* a menos que na etapa anterior. Ou seja, após a primeira etapa haverá $N - 1$ *clusters*. Esse processo é repetido até que alcance apenas um único *cluster*, após $N - 1$ etapas, ou alcance um limiar de distância máximo, determinando o quão diferentes dois *clusters* podem ser [15].

A similaridade também foi usada com o objetivo de diminuir a influência negativa de clientes maliciosos ou com resultados ruins de treinamento. Wang et al. [105] propuseram o uso de similaridade de cossenos para executar uma agregação adaptativa e amenizar a influência negativa dos clientes. Nessa mesma perspectiva, Duan et al. [31] propuseram agrupar os clientes a partir da avaliação indireta da similaridade dos dados considerando uma nova métrica denominada de distância euclidiana de similaridade de cosseno decomposta.

Por sua vez, Xie et al. [111] propuseram avaliar a similaridade entre os clientes e enviar o modelo global personalizado para cada cliente ponderando o relacionamento entre eles, para isso foi usado a métrica Similaridade de Gradiente Normalizado Softmax [111].

A principal contribuição desta tese com a proposta do MoSimFeL é enviesar a seleção de clientes considerando seus históricos de contribuição em treinamentos anteriores através de métricas de similaridade entre modelo-modelo e de similaridade entre gradiente-modelo. Especialmente em relação ao termo gradiente-modelo, é importante explicar que essa métrica analisa a similaridade entre o gradiente resultante do treinamento local de um cliente e o modelo global resultante do mesmo ciclo de treinamento. O gradiente de um treinamento local é indiretamente um novo modelo de AM, mesmo que treinado a partir do enviesamento exclusivo da base de dados de um único cliente. Na similaridade gradiente-modelo, há uma relação de causa e efeito entre os dois parâmetros avaliados, como já explicado. Por sua vez, na similaridade modelo-modelo, trata-se da avaliação de dois modelos distintos quaisquer, tais como dois modelos diferentes treinados por dois servidores centrais distintos e com um conjunto de clientes também distintos.

Para a avaliação de similaridade entre modelo-modelo, este pesquisa optou pela métrica CKA, considerando os trabalhos mais recentes da literatura que comprovam a eficiência da métrica na avaliação de similaridade entre modelos de rede treinados. Por sua vez, na avaliação de similaridade entre gradiente-modelo, esta pesquisa optou, em sua solução final, pelo uso da métrica de distância euclidiana, mas também a distância de Manhattan também foi analisada.

Dessa maneira, nas próximas subseções, apresenta-se a fundamentação conceitual sobre a avaliação de similaridade por CKA, distância euclidiana e distância de Manhattan. Nessas subseções, trabalhos importantes dentro do respectivo contexto que fundamentam essa tese também são abordados.

2.2.1 Similaridade Modelo-Modelo

A avaliação de modelos de AM por CKA foi definida em [55]. Para definir a similaridade CKA entre dois modelos de AM treinados M_1 e M_2, definimos $X \in \mathbb{R}^{m,p1}$ como sendo a representação de uma matriz de ativação de p_1 neurônios de M_1 sob m exemplos, tal que, $m, p_1 \in \mathbb{N}$. De forma análoga, $Y \in \mathbb{R}^{m,p_2}$ representa a matriz de ativação de M_2 para a mesma base de dados com m exemplos [39, 55, 86].

Dessa maneira, Kornblith et al. [55] deduzem que é possível definir $K = XX^T$ e $L = YY^T$ como sendo o produto escalar (m, m) entre a representação de ativação da i^{th} e da j^{th} amostras sob os modelos M_1 e M_2.

Assim, é possível definir o critério de Independência de Hilbert-Schmidt, entre K e L e indiretamente entre M_1 e M_2, como $HSIC_O$ descrito por 2.1 [55].

$$HSIC_O(K, L) = \frac{1}{(m-1)^2} tr(X'Y'), \tag{2.1}$$

onde $X' = XH,\ Y' = YH$ e $H_m = I_m - \frac{1}{m} 11^T$.

Na Equação 2.1, a função $tr(W)$ é o traço da matriz, tal que $tr(W_{qxq}) = \sum_{i=1}^{q} w_{ii}$. Através de $HSIC_0$, define-se CKA a partir da Equação 2.2 [55, 86].

$$CKA(K, L) = \frac{HSIC_O(K, L)}{\sqrt{HSIC_O(K, K) HSIC_O(L, L)}} \tag{2.2}$$

Todavia, Nguyen et al. [86] destacaram que o cálculo da Equação 2.2 necessita que as matrizes de ativação de todas as amostras sejam mantidas na memória, tornando-se um desafio à medida que os valores de p_1, p_2 e m aumentam. Dessa maneira, os autores analisaram o CKA a partir de lotes de amostras, reduzindo o uso da memória, conforme descrito pela Equação 2.3 [86].

$$CKA_{lote}(K, L) = CKA_{lote}(X_i X_i^T, Y_i Y_i^T)$$

$$CKA_{lote}(K, L) = \frac{\frac{1}{k} \sum_{j=1}^{k} HSIC_1(X_i X_i^T, Y_i Y_i^T)}{\sqrt{\frac{1}{k} \sum_{j=1}^{k} HSIC_1(X_i X_i^T, X_i X_i^T)} \sqrt{\frac{1}{k} \sum_{j=1}^{k} HSIC_1(Y_i Y_i^T, Y_i Y_i^T)}} \tag{2.3}$$

onde X_i e Y_i são as matrizes de ativação do i^{th} de n amostras de um lote de dados, tal que $X_i \in \mathbb{R}^{m,p_1}$ e $Y_i \in \mathbb{R}^{m,p_2}$ [86].

Por fim, Nguyen et al. [86] definiram $HSIC_1$ através da Equação 2.4, onde $\ddot{K}$ e $\ddot{L}$ são obtidos configurando as diagonais das matrizes K e L como zero [86].

$$HSIC_1(K,L) = \frac{1}{n(n-3)}\left(tr(\ddot{K}\ddot{L}) + \frac{1^T\ddot{K}11^T\ddot{L}1}{(n-1)(n-2)} - \frac{2}{(n-2)}1^T\ddot{K}\ddot{L}1\right) \tag{2.4}$$

Assim, avaliando a similaridade entre todas as camadas de M_1 com todas as camadas de M_2, aplicando as Equações 2.2 ou 2.3 é possível definir uma matriz de similaridade cruzada entre todas as camadas dos dois modelos denominada de R_{M_1,M_2}, tal que $r \in R_{M_1,M_2}$, $r \in [0,1]$ e h é a ordem da matriz R_{M_1,M_2}. Além disso, quanto mais r é próximo de 1, mais os modelos M_1 e M_3 são similares e o inverso também é verdadeiro.

Considerando que R representa a combinação de relação entre todas as camadas de M_1 e M_2, opta-se por representar essa estrutura de dados de duas dimensões em um valor escalar. Para tal, é levantado as seguintes propostas: norma da matriz ou norma da diagonal secundária, respectivamente descritas nas Equações 2.5 e 2.6.

$$CKAN_{modulo}(M_1,M_2) = |R_{M_1,M_2}| \tag{2.5}$$

$$CKAN_{secundaria}(M_1,M_2) = |r_{i,j}|, \forall j = (h-i+1) \tag{2.6}$$

A fim de exemplificar a funcionalidade do *CKA* e definir a melhor forma de transformar a matriz de similaridade em um valor escalar, é definido um experimento comparando os resultados das Equações 2.5 e 2.6.

No experimento, usa-se uma arquitetura CNN e, a partir dessa arquitetura, modelos distintos foram treinados com diferentes subconjuntos de dados da mesma base de dados MNIST [59].

Ao todo, três cenários foram definidos e, em cada um dos cenários, há o treinamento de dois modelos, denominados didaticamente de $M_{x,1}$ e $M_{x,2}$, em que x representa o identificador do cenário. No treinamento, não foi aplicada a técnica de AF, pois a avaliação de similaridade proposta independe da estratégia de treinamento.

No *Cenário I*, o modelo $M_{1,1}$ foi treinado com 90% de dígitos entre $[0, 4]$ e 10% entre $[5, 9]$. Por sua vez, o modelo $M_{1,2}$ foi treinado apenas com 90% dígitos $[5, 9]$ e 10% entre $[0, 4]$. Ou seja, os modelos foram treinados com dois conjuntos de classe quase que distintos.

No *Cenário II* a distribuição de dados do modelo $M_{2,1}$ manteve uma distribuição de 75% dos dados para o conjunto de dígitos entre $[0, 4]$ e 25% para o conjunto de dígitos entre $[5, 9]$. Por sua vez, o modelo $M_{2,2}$ foi treinado com um conjunto de 25% dos dados entre $[0, 4]$ e 75% para o conjunto de dígitos entre $[5, 9]$.

No *Cenário III*, o conjunto de dados foi equalizado e os dois modelos ($M_{3,1}$ e $M_{3,2}$) mantém uma distribuição de dados igualmente distribuída entre os dígitos $[0, 9]$, ou seja, 50% entre $[0, 4]$ e 50% entre $[5, 9]$.

Em todos os cenários, foram usadas 27100 amostras e, dentro de um mesmo conjunto, todas as classes tinham a mesma quantidade de amostras.

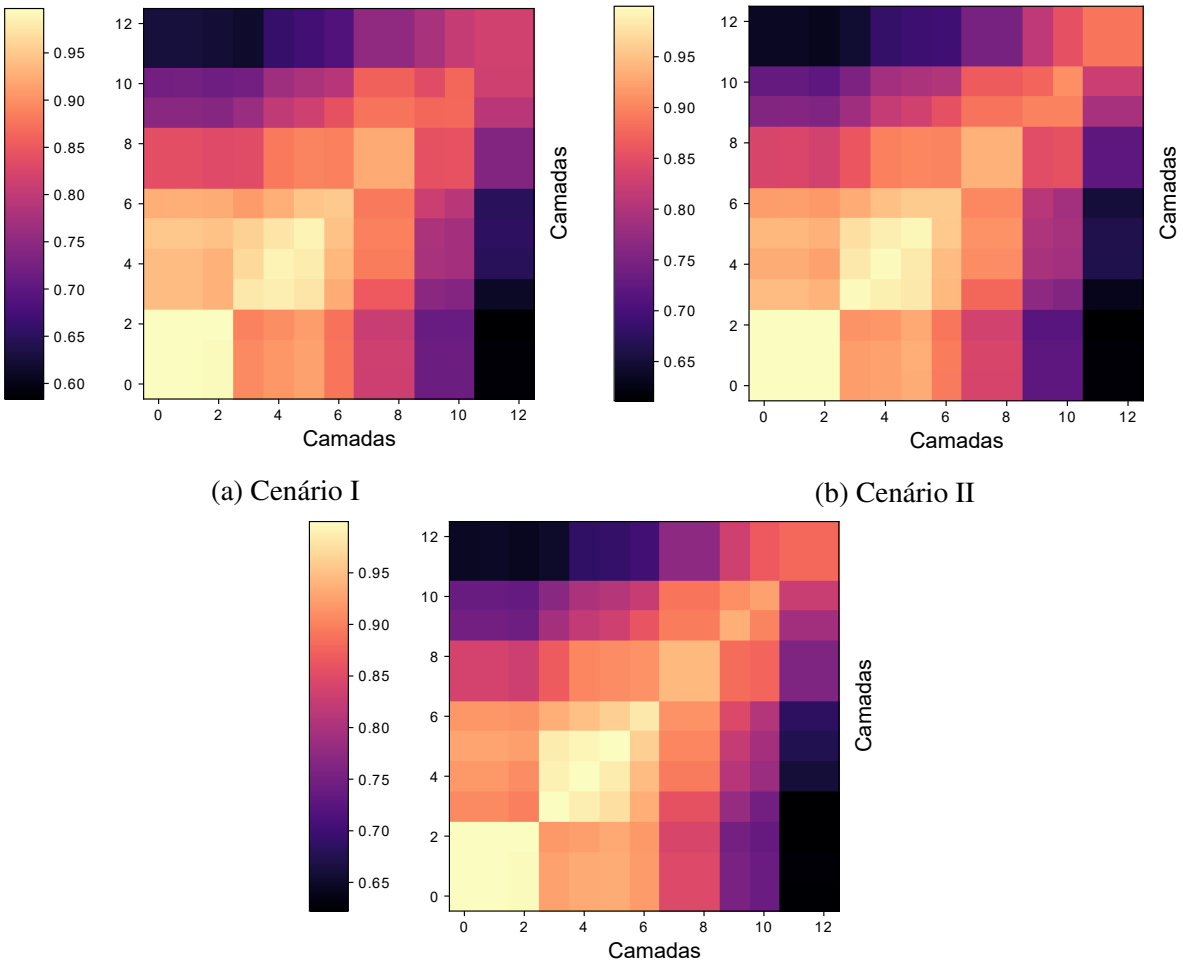
Os modelos foram treinados com 15 épocas e tamanho do lote 32. O número de épocas e o tamanho do lote foram definidos a partir de uma avaliação prévia, analisando treinamentos isoladamente e concluindo que a escolha desses parâmetros garantia uma boa convergência. Por fim, considerando a natureza não determinística do treinamento, cada cenário foi treinado 20 vezes, e os resultados discutidos a seguir representam a média dessas execuções. Observou-se que todos os treinamentos convergiram, independentemente do cenário, graças à base de dados, à arquitetura e aos parâmetros propostos.

Na Figura 2.2[1], exemplifica-se as matrizes de similaridade dos três cenários na análise de CKA entre os modelos $M_{x,1}$ e $M_{x,2}$ de uma das 20 execuções. Comparando a escala de cores da matriz do *Cenário I* e da matriz do *Cenário II*, é difícil identificar visualmente as diferenças. O mesmo acontece quando compara-se visualmente as matrizes do *Cenário II* e *III*. Todavia, comparando os extremos, *Cenários I* e *III*, identifica-se visualmente que a matriz do *Cenário III* está, em geral, mais clara, ou seja, aparenta ter uma similaridade maior do que o *Cenário I*.

Na Tabela 2.1, apresenta-se, para cada um dos cenários, o valor escalar para as Equações 2.5 e 2.6.

Considerando os resultados da Equações 2.6 e 2.5, conclui-se que, à medida que a distri-

[1]Os gráficos de avaliação de similaridade foram plotados a partir do desenvolvimento de uma rotina própria de programação com inspiração na biblioteca Subramanian [99].



(a) Cenário I

(b) Cenário II

(c) Cenário III

Figura 2.2: Matrizes de similaridade CKA dos *Cenários I, II e III*.

buição das bases de dados de $M_{x,1}$ e $M_{x,2}$ é equalizada, a similaridade aumenta. Portanto, na análise de modelos de AM de mesma arquitetura, a avaliação de similaridade entre os modelos pode ser usada para inferir diretamente à similaridade entre os conjuntos de dados de treinamento. Para o MoSimFeL, essa premissa é utilizada para analisar a similaridade entre os modelos para avaliar a base de dados dos clientes, sem infringir o princípio de privacidade.

Como ambas as equações demonstraram, através de um escalar, o mesmo comportamento de aumento de similaridade em decorrência da equalização da base de dados, a definição da melhor estratégia deve ser determinada pela aplicação. A Equação 2.5 considera a relação cruzada de todas as camadas do modelo $M_{x,1}$ com todas do modelo $M_{x,2}$ e não apenas os pares das camadas de cada modelo com a mesma profundidade. Por sua vez, a Equação 2.6 enfatiza a comparação entre as camadas do mesmo nível.

	Norma Módulo	**Diagonal secundária**
Cenário I	$10,783$	$2,786$
Cenário II	$10,894$	$2,814$
Cenário III	$10,954$	$2,837$

Tabela 2.1: Resultados da norma e do norma da diagonal secundária para a matriz resultante da análise do CKA para os *Cenários I, II e III*.

A vantagem da Equação 2.5 é reconhecer que uma camada pode ser compensada por outra, visto que ele considera a comparação de uma camada com todas as demais. Por outro lado, ao ser mais restrita, a Equação 2.6 reconhece que níveis diferentes também possuem pesos diferentes na tarefa do modelo, logo, comprar uma camada inicial com uma final passará uma falsa similaridade.

Acredita-se que estudos mais aprofundados podem encontrar métodos mais eficientes de avaliação da matriz de similaridade, incluindo o objetivo de definir um valor escalar para tal. Uma sugestão é avaliar a diagonal secundária e seu entorno mais próximo, como um meio-termo entre as Equações 2.6 e 2.5. Outra sugestão é ponderar os níveis de camada a fim de valorizar mais a semelhança entre as camadas finais do que as iniciais, pois, em uma rede de aprendizagem profunda, as camadas finais, mais próximas à saída da rede, representam mais o conhecimento do que as iniciais. No entanto, ao menos para o escopo desta tese, ambas as equações são suficientes, de forma que a execução desse estudo será apontada entre os trabalhos futuros.

A análise da Equação 2.2 pode ser um desafio significativo para uma aplicação com requisitos de privacidade de dados, mediante a necessidade de compartilhar os modelos de AM treinados dos servidores centrais entre diferentes dispositivos, o que favorece ataques para inferir os dados dos usuários. Além disso, as restrições impostas podem dificultar o compartilhamento de bases de dados necessárias para o cálculo do CKA. Portanto, a depender da rigidez dos requisitos, o uso da técnica CKA pode ser um desafio significativo em cenários de AF. Nesse sentido, na Seção 4.2 propõe-se o FCKA, como um protocolo para execução do CKA e, consequentemente, *CKAN* de forma descentralizada e com mitigação do compartilhamento dos dados no contexto de AF.

Enfatiza-se que a avaliação de similaridade por CKA é determinística, mas varia conforme a entrada do conjunto de dados de avaliação definido. Ou seja, para um mesmo conjunto de dados e os mesmos modelos, a similaridade definida pelo CKA é sempre a mesma. Todavia, ao alterar a base de dados, o resultado de similaridade também sofrerá alterações. Assim, é importante que D_{cka} represente todo o domínio em que os modelos serão aplicados para que o CKA represente a similaridade de forma coerente.

Evidentemente, não é possível formar D_{cka} a partir das amostras exatas às quais o modelo será submetido, principalmente em aplicações com AF, devido ao princípio de privacidade. Todavia, é preciso reconhecer que nem todos os dados de uma aplicação são necessariamente privados ou que todos os clientes são irredutíveis quanto a esse requisito [37, 119, 128]. É possível definir estratégias para inferir um conjunto de dados que represente esse domínio com um grau mínimo de fidelidade, sem comprometer o princípio da privacidade. Entre essas estratégias, destacam-se o uso de dados artificiais, a criação de uma base de dados própria no servidor central, a disponibilização de dados por clientes que optem por abdicar da privacidade, a oferta de recompensas a clientes que disponibilizem parcialmente seus dados e o uso de dados históricos, sem valor para aplicação ou para os clientes, mas que ainda representem o domínio.

Apesar de variar conforme o conjunto de dados, o comportamento de variação do resultado das Equações 2.6 e 2.5 a medida em que os modelos são treinados com bases de dados mais próximos entre si, como observado na Tabela 2.1, é o mesmo. Isso é evidenciado pois, em todas as vezes em que executou a simulação do cenário, foi utilizado uma base de dados diferente (respeitando a distribuição imposta) e em todas as vezes esse comportamento se repetiu. Ou seja, em aplicações de treinamento federado, não é restritamente necessário a utilização da base de dados dos clientes para análise de similaridade, caso seja necessário apenas comparar a similaridade relativa entre diferentes bases.

2.2.2 Similaridade Gradiente-Modelo

A similaridade entre os modelos (M_1 e M_2) treinados a partir de uma mesma arquitetura de AM também pode ser definida através do gradiente que diferencia os modelos, quando há uma relação de treinamento entre eles. O gradiente representa uma parcela de quanto o modelo M_1 deve ser atualizado até se igualar a M_2, visto que o gradiente é a base de atualização

dos pesos e bias dos neurônios durante o treinamento de um modelo de AM. Considerando o FedAVg, essa parcela é a proporção do volume de dados usados no treinamento local de um cliente em relação ao volume de dados total usados por todos os clientes durante o mesmo treinamento [111].

No contexto da AF, para avaliar a similaridade entre um modelo resultante de um treinamento local de um cliente n e um modelo global definido pelo servidor central no mesmo ciclo de treinamento, é possível utilizar o gradiente do treinamento local (M_1), denominado de Δ_{n,M_1,M_2}. Dessa maneira, avaliando a distância entre M_2 e Δ_{n,M_1,M_2}, denominada de $SD(\Delta_{n,M_1,M_2}, M_2)$, é possível inferir diretamente a similaridade entre M_1 e M_2, de forma que, quanto maior a distância, menor a similaridade. Para calcular $SD(\Delta_{n,M_1,M_2}, M_2)$, é possível usar a distância euclidiana (Equação 2.7) ou a de Manhattan (Equação 2.8).

$$SD(n, M) = \sqrt{\sum_{i=j}(n_i - m_j)^2} \tag{2.7}$$

$$SD(n, M) = \sum_{i=j}(|n_i - m_j|) \tag{2.8}$$

Essas métricas já foram abordadas anteriormente na avaliação de similaridade entre treinamentos locais de clientes, com o que poderia ser denominado, seguindo o padrão dessa de tese, de similaridade gradiente-gradiente. Nesse contexto, segundo Briggs et al. [15], a distância euclidiana é usada para avaliar a similaridade entre vetores, a distância de Manhattan é adequada para a avaliar a similaridade entre vetores esparsos e a distância do cosseno indica apenas o quão próximo dois vetores apontam na mesma direção e é invariante aos efeitos de escala.

Para avaliar qual a melhor métrica de distância $SD(\Delta_{n,M_1,M_2}, M_2)$ é proposto um experimento a partir de uma aplicação de classificação de imagens (com uma arquitetura CNN) que é executada usando a base de dados MNIST com dígitos manuscritos. O cenário possui 110 clientes e 1 servidor central que executam o treinamento federado, em que todos os clientes são selecionados para o treinamento local. Cada cliente detém uma fatia de amostras da base de dados. A distribuição de dados no cenário é feita enviesadamente de forma que 10 clientes têm uma base de dados equilibrada com o mesmo número de amostras para cada

	Manhattan	**Euclidiana**
Clientes Enviesados	$180331,50$	$791,33$
Clientes Não-Enviesados	$92234,56$	$577,81$

Tabela 2.2: Distância de Manhattan e Euclidiana considerando o treinamento dos clientes enviesados e não-enviesados.

classificador e 100 clientes tem um conjunto de dados desbalanceado e enviesado para um classificador, de forma que, 10 clientes são enviesados para cada um dos classificadores entre $[0-9]$.

Os clientes enviesados detêm de 2000 amostras representando seu classificador principal em detrimento de 40 amostras para cada um dos classificadores restantes. Por sua vez, os clientes com base de dados equilibrada detêm de 236 amostras para cada um dos classificadores. Portanto, cada cliente possui 2360 amostras.

Ao todo, cada cliente executou o treinamento local por 15 épocas em cada ciclo e o modelo foi treinado por 15 ciclos seguidos.

Considerando a distribuição panorâmica de dados no cenário, o conjunto total de amostras está balanceado, pois cada classificador tem o mesmo número de amostra no cenário. Assim, considerando a distribuição de dados, a arquitetura do modelo e os parâmetros definidos, espera-se que o treinamento federado garanta um aprendizado eficiente do modelo global.

Com o modelo treinado, para todos os clientes calculou-se a distância a partir das Equações 2.7 e 2.8 encontrando os resultados apresentados na Tabela 2.2. Nessa tabela, os resultados apresentados representam as médias das distâncias encontrados pelo conjunto de clientes com o mesmo enviesamento.

Através da Tabela 2.2 é possível identificar que os clientes com base de dados enviesadas tiveram uma distância menor do que os clientes com base de dados equalizada em todas as métricas de distância avaliadas.

Dessa maneira, considerando a base de dados e a aplicação propostas, todas as métricas são consideradas adequadas, pois a distância para o conjunto de clientes não-enviesados foi menor do que para o conjunto enviesado. Ou seja, em ambas as métricas, a distância para o conjunto de clientes não-enviesados é menor do que para o conjunto enviesado. Isso ocorre

porque o modelo treinado pelos clientes não enviesados certamente ficará mais próximo do modelo global, devido à base de dados desses clientes representar o mesmo padrão da soma de todas as bases de todos os clientes

Assim como na análise de similaridade modelo-modelo, é possível melhorar a análise de similaridade de gradiente-modelo considerando a importância de cada camada na arquitetura do modelo. Por fim, um estudo comparativo entre técnicas de similaridade pode ser expandido a fim de avaliar a eficiência dessas técnicas em diferentes contextos, como a Similaridade de Gradiente Normalizado Softmax [111].

2.3 Não estacionariedade

A aprendizagem em ambientes não estacionários lida com o constante desvio de conceito de aprendizagem e por isso as técnicas de aprendizagem precisam estar adaptadas a características estatísticas mudam ao longo do tempo [29]. Esse fenômeno pode ter diferentes causas, mas é possível resumi-las em dois grupos: variação no ambiente ou variação nos dispositivos [30].

No contexto de AF, o ambiente não-estacionário é um desafio ainda maior visto que essa técnica de treinamento é sensível à heterogeneidade de dispositivos e a distribuição de dados não-IID, como já explicado nesta tese. Tratando-se de ambientes com mobilidade, o desafio é ainda maior, mediante o dinamismo que essa característica promova no ambiente. Logo, a AF pode não ser uma solução viável para esses cenários, caso os algoritmos de coordenação de treinamento não se adaptem a esse ambiente [20].

Apesar da importância de prever a não-estacionariedade, principalmente por ser uma caraterística comum de se encontrar em aplicações modernas, há ainda poucos trabalhos na literatura que investigam esse conceito dentro do contexto de AF. A maioria dos trabalhos encontrados na literatura, tal como os apresentados nesta tese, abordam problemas e apresentam soluções considerando ambientes estacionários, mesmo quando na descrição do problema ou na exemplificação de cenários são descritos ambientes com características não estacionárias [20, 36].

A fim de garantir robustez de AF em ambientes não estacionários, Casado et al. [20] propôs transformar o FedAvg em um algoritmo assíncrono de forma que os clientes podem

executar o treinamento localmente à medida que necessitam atualizar o modelo. Por sua vez, o servidor central decide se a contribuição desse cliente favorece ou não o progresso de aprendizagem do modelo. Caso favoreça, o servidor central agrega os parâmetros enviados pelo cliente e atualiza o modelo global. Com isso seu algoritmo se adapta continuamente ao desvio de conceito que dinamicamente ocorre na aplicação.

A priori o único desafio relacionado ao desvio de aprendizagem seria identificar a necessidade de um novo treinamento a partir de um novo conjunto de dados. Todavia, além de identificar, é preciso analisar como o modelo deve progredir com sua aprendizagem. Uma solução preliminar é iniciar o treinamento a partir da definição aleatória dos neurônios, ou seja, sem uma bagagem de lições anteriores. Todavia, a tarefa de iniciar um novo modelo é mais custosa do que o retreinamento de modelos com aprendizagens anteriores. Isso porque um modelo recém inicializado define aleatoriamente seus parâmetros, tais como pesos e bias. Por sua vez, modelos anteriormente treinados com lições passadas já possuem um enviesamento dos parâmetros no ponto de partida do treinamento.

Para exemplificar, em um experimento de simulação, define-se a mesma arquitetura da Seção 2.2.1. Um modelo dessa arquitetura é treinado com a base de dados MNIST considerando subconjuntos distintos e momentos diferentes. O número de épocas de cada treinamento não é fixo e o modelo é treinado até a acurácia mínima de 0.96. Por mim o tamanho do lote de treinamento é 32.

Ainda sobre o experimento, compara-se dois cenários. No primeiro cenário, *Cenário IV*, um modelo é treinado duas vezes. Na primeira vez é disponibilizado 5421 amostras de cada classe entre $[0-4]$ e nenhuma amostra entre as classes $[5-9]$. Com a conclusão do primeiro, inicia-se o segundo treinamento com 5421 amostras para todas as classes, ou seja, entre $[0-9]$. A proposta deste segundo cenário é mensurar o progresso da aprendizagem do modelo durante o segundo treinamento.

Por sua vez, no *Cenário V*, um modelo é treinado apenas uma vez com o conjunto total de 5421 amostras para cada classe entre $[0-9]$. Ou seja, esse cenário representa um treinamento de um novo cenário a partir de um modelo completamente novo.

O experimento foi repetido 50 vezes. Os resultados mostraram que em 49 vezes, 98%, o segundo treinamento do *Cenário IV* alcançou a acurácia mínima em menos épocas do que o *Cenário V*.

Conclui-se, ao menos nesse exemplo, que, caso uma aplicação mantenha uma instância de modelo em treinamento, é melhor retreinar esse modelo para adaptar-se a um novo cenário, do que iniciar um treinamento de um novo modelo. Obviamente essa conclusão não se repete para um modelo que muda completamente de aprendizagem, alterando seu domínio de atuação. Por exemplo, uma aplicação que inicialmente classifica imagens de números e passa a classificar letras. Nesse caso, ao menos, considerando técnicas tradicionais de treinamento, o arranjo dos parâmetros resultante de um treinamento anterior é o mesmo que a definição de parâmetro aleatória para um novo modelo. Mesmo assim, ainda é possível avaliar técnicas mais inteligentes como a transferência de treinamento para que o aprendizado passado ainda seja aproveitado no processo de aprendizado das novas lições [92].

Capítulo 3

Revisão da Literatura

Neste capítulo, os principais trabalhos da literatura relacionados à mobilidade, não-estacionariedade e similaridade são abordados. Diferente do Capítulo 2, em que também diversos artigos fora revisados, neste capítulo é enfatizado a revisão da literatura mais próxima a temática desta tese, considerando a apresentação dos desafios, soluções e uma discussão sobre os trabalhos. A apresentação da revisão da literatura será norteada pela exemplificação de diferentes cenários com o intuito de elucidar a proposta de novas soluções para os desafios apresentados. Ademais, a apresentação da revisão da literatura é norteada pela exemplificação de diferentes cenários com o intuito de elucidar a proposta de novas soluções para os desafios apresentados.

Assim, esse capítulo é dividido nas seguintes seções: na Seção 3.1, contextualiza-se um cenário exemplo, com a citação de aplicações e o levantamento de requisitos e características dessas aplicações; na Seção 3.2, analisa-se trabalhos com soluções baseadas em Aprendizagem Federada Personalizada e Aprendizagem Federada Multi-tarefas; na Seção 3.3, analisa-se trabalhos com soluções baseadas em propostas de arquiteturas; na Seção 3.4, avalia-se trabalhos relacionados AF e o desvio de aprendizagem; Por fim, na Seção 3.5, discute-se a conclusão dessa revisão.

3.1 Cenários de Aplicações de AF

O desafio de implementação de uma aplicação de AF começa desde o planejamento da arquitetura, que define como os dispositivos se conectam e interagem na aplicação. Claramente,

esse é um desafio para qualquer aplicação em um cenário moderno, independentemente da estratégia de treinamento. Todavia, esse desafio é evidenciado no AF, pois os recursos computacionais que executam o treinamento do modelo de AM são os clientes que, em muitos casos, também são os usuários finais da aplicação [89].

Para exemplificar o desafio de implementação de uma aplicação de AM, na Fig. 3.1, ilustra-se um cenário de uma cidade com veículos, equipamentos de IoT e humanos conectados a diferentes aplicações e em diferentes redes. Como mencionado introdutoriamente no Capítulo 1, alguns desses dispositivos representam diretamente o usuário por serem um canal direto de interação com ele, como os *smartphones*, enquanto outros são sensores ou atuadores inteligentes que compõem a aplicação, como semáforos que definem a sinalização com base em sensores que contam veículos trafegando em uma determinada via. Nesse último exemplo, o usuário da aplicação é o motorista que se beneficia da sinalização eficiente, mas os periféricos (semáforos e sensores) se comunicam em prol da aplicação.

Nessas aplicações, os dispositivos precisam ser confiáveis e estarem disponíveis quando requisitados. Isso é um desafio visto que ocasionalmente se desconectam da rede por diversos motivos, como falta de energia (ou nível de bateria baixo) e rede inacessível [108].

Nesse contexto, alguns dispositivos ainda possuem um agravante para manter sua conexão: a mobilidade. A mobilidade é um desafio pois, à medida que trafegam, mudam a conexão entre micro estações base (no inglês, *micro state unit* - MSU) ou pontos de acesso de borda (no inglês, *edge access points* - EAPs) em busca de melhor qualidade de canal de comunicação.

Por exemplo, em redes veiculares, os veículos produzem e consomem uma grande quantidade de dados, através de sensores como câmeras e GPS. Esses dados podem ser explorados no treinamento de algoritmos de AM [68]. A mobilidade intensa dos veículos pode facilmente provocar a desconexão e a deterioração da qualidade da comunicação entre os dispositivos. Além disso, os veículos podem alterar a conexão entre diversos servidores de borda, instalados em unidades de beira de estrada (no inglês, *road state units* - RSUs), à medida que alcançam a área de cobertura de diferentes estações de transmissão, por meio de comunicação sem fio. Essas trocas impactam na continuidade de um serviço fornecido na rede, incluindo aplicações de AF [68]. Um exemplo de aplicação veicular que está sendo implementada sobre a borda e com auxílio de AM é a direção autônoma que emula compor-

Fonte: Adaptado de Macedo et al. [75]

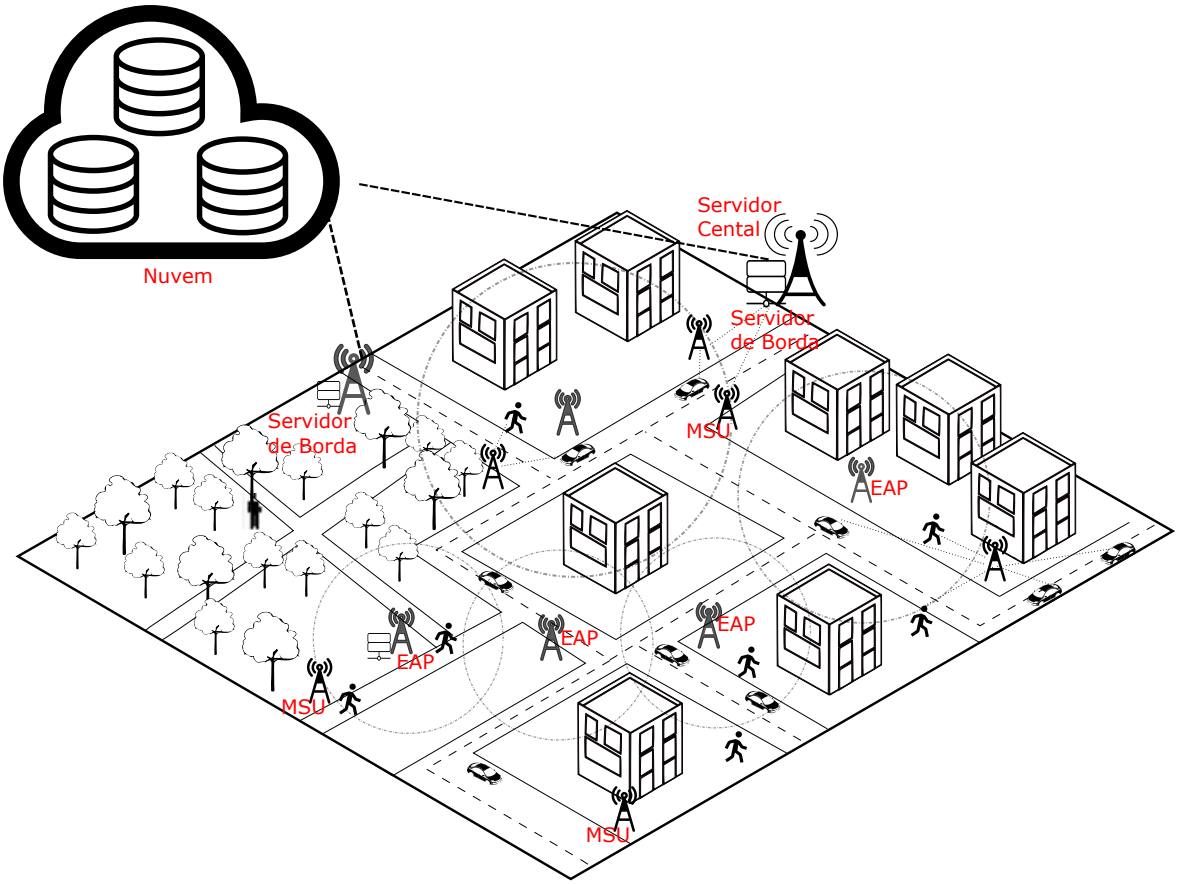


Figura 3.1: Ilustração de celulares inteligentes e veículos se locomovendo e migrando de servidor de borda e servidor central em uma cidade.

tamentos humanos na ação de dirigir veículos [119].

Outro exemplo é o cenário com celulares inteligentes ou dispositivos de IoT. Os limitados recursos energéticos, a restrição de disponibilidade de recursos computacionais e a mobilidade provocam a alteração de rede, incluindo a mudança de provedores e tecnologias de rede sem fio (*Wi-Fi*, *Bluetooth*, 4G, 5G e outras). Essa mudança visa garantir uma melhor qualidade de conexão do usuário com a rede e provoca mudanças nos servidores de borda conectados (representados pelos EAPs na Fig. 3.1) e na mudança de servidores centrais para aplicações de AF, quando são instalados nos servidores de borda. Ou seja, a mobilidade não apenas pode causar a desconexão de um dispositivo de um domínio de rede, mas também pode levar à conexão em novos domínios.

Outro desafio no cenário exemplificado é viabilizar a especialização do modelo para

aprendizagem de nuances específicas de um domínio do problema, garantindo um bom desempenho da aplicação, ao mesmo tempo em que mantém a capacidade de generalização para tratar a diferenças entre os domínios. A diferença entre os domínios de uma mesma aplicação pode ser tão acentuada a ponto de conquistar a atenção de pesquisadores e desenvolvedores que definiram o termo multidomínios para se referir a esses cenários. Por exemplo, em aplicações de detecção de objetos em imagens que são capturadas a partir de câmeras com especificações extremamente distintas. Para desenvolver aplicações para cenários multidomínios é possível aplicar técnicas de Aprendizagem Multi-tarefas (AMT) e Aprendizagem Federada Personalizada (AFP) [34]. A AMT tem o objetivo de treinar um único modelo para resolver diversas tarefas, explorando semelhanças e divergências entre as soluções das tarefas. No caso, cada domínio também pode ser considerado uma tarefa. Por sua vez, a AFP mantém a personalização de diversas instâncias de modelos treinados, um para cada domínio. Todavia, apesar da distinção das instâncias, o treinamento dos modelos personalizados tem características em comum, como o uso de um subconjunto de dados similar a todas as bases de treinamento ou a mesma arquitetura.

A adesão a um único modelo na aplicação para garantir a especialização e generalização pode ser um grande desafio, pois um único modelo pode ser incapaz de assimilar as especificidades de todos os domínios do problema, como é proposto na AMT. Além disso, a AMT requer que os domínios tenham alguma relação entre si que possa ser assimilado durante a aprendizagem para compensar o desafio de múltiplos domínios.

Um exemplo desse desafio é a identificação de carteiras de identidade de pessoas a partir do processamento de imagens. Cada país, ou até mesmo estados de um mesmo país, possui padrões de documentos de identificação distintos entre si, sendo comum até mesmo mais de um tipo de documento em uma mesma região. No Brasil, por exemplo, é possível que uma pessoa se identifique com a carteira de motorista, com o passaporte ou com a carteira de identidade. Considerando a diversidade de documentos de identificação disponíveis no mundo, como diferentes diagramações, modelos, codificações de caracteres, cores e tipos de papel, o treinamento de um modelo de AM capaz de interpretar as informações de todas as classes desses documentos é uma tarefa difícil. Essa tarefa é árdua mesmo dispondo de uma base de dados vasta que contemple todas as classes para o treinamento, pois a complexidade da arquitetura da rede profunda cresce a ponto de inviabilizar o aprendizado.

Por sua vez, em aplicações nas quais diferentes domínios surgem a partir da temporalidade, é possível treinar modelos distintos para cada intervalo de tempo em que um domínio se sobrepõe e trocar os modelos conforme a variação temporal. Um exemplo típico desse cenário são os veículos autônomos que trafegam a partir do processamento de imagens e da identificação de faixas delimitadoras de vias. Devido à mudança de iluminação durante o dia, é interessante analisar modelos que sejam especialistas no processamento de imagens considerando a luminosidade, viabilizando uma alta acurácia na identificação dos padrões de imagem.

A proposta de definir diferentes modelos com base na especificidade de um domínio também é interessante para problemas cujos domínios são definidos por requisitos de negócios da aplicação. Por exemplo, uma aplicação para auxiliar na avaliação do comportamento consumista de um cliente é implementada em diferentes lojas de comércio virtual. Nessa aplicação, o comportamento de navegação na internet do potencial cliente é avaliado constantemente para modificar o fluxo de navegação no comércio virtual e influenciar na decisão de compra do cliente. Neste exemplo, diferentes modelos podem ser mantidos por diferentes grupos de lojas devido à confidencialidade das decisões comerciais e às particularidades de cada tipo de comércio. Ou seja, os domínios são definidos a partir de requisitos de negócios, como esperar que o comportamento dos clientes e a definição do melhor fluxo de navegação para favorecer o consumo sejam diferentes em um comércio virtual voltado para ferramentas em comparação com uma loja voltada para vestuário de luxo. Nesse contexto, a aplicação e até mesmo a arquitetura do modelo de AM treinado podem ser os mesmos, porém as bases de dados usadas no treinamento são distintas.

Outro desafio enfrentado no cenário da Fig. 3.1 é o desvio de aprendizagem. O desvio de aprendizagem acontece quando o modelo de AM já treinado perde a eficiência na execução de sua tarefa devido a mudança de características dos dados em que a aplicação é submetida (conhecido por desvio de dados) ou mudança de classificação nos dados (desvio de conceito). O desvio de aprendizagem acontece esporadicamente ou periodicamente por diversas causas, como mudança de comportamento dos usuários da aplicação, falhas em sensores, falhas em atuadores e mudanças do ambiente [19, 20, 50, 51].

Para exemplificar, retoma-se uma aplicação do Capítulo 1 cujo objetivo é identificar o comportamento de consumo de um cliente ao visitar lojas físicas e avaliar a tendência de

consumo com base no tempo de permanência do cliente em diferentes setores de lojas. Nessa aplicação, a mudança nas condições climáticas, como o início de uma chuva na região onde o cliente está localizado, pode interferir rapidamente em seu comportamento, alterando seu interesse em consumo. Nesse contexto, a aplicação precisa identificar rapidamente a mudança de interesse do cliente para atualizar o modelo e enfrentar os novos desafios.

Ainda relação ao desvio de aprendizagem, a avaliação da eficiência do modelo pode mudar drasticamente com a participação de novos usuários em um domínio específico da aplicação, alterando a avaliação coletiva do modelo. Se a avaliação do modelo pelos novos usuários apontar para uma insatisfação, o servidor central pode recorrer à atualização do aprendizado por meio de um novo treinamento.

Esse cenário é comum em aplicações em que o histórico de mobilidade de um indivíduo influencia diretamente na eficiência da aplicação. Por exemplo, considerando a aplicação de identificação de faixas para veículos autônomos, automóveis que trafegam em uma região sem asfalto e com estradas de terra podem sujar as lentes de suas câmeras, que captam imagens das faixas. Dessa forma, as imagens dessas câmeras formam um conjunto de dados diferente dos veículos com lentes limpas. Outra região de tráfego que contenha asfalto limpo, resultando em alta nitidez e diferença de coloração entre o asfalto e a faixa, deve ter um modelo específico para reconhecimento que leve em consideração as características dessa região e pressuponha que uma parte significativa dos veículos que trafegam nela mantenha as lentes de suas câmeras limpas. No entanto, é possível que, na percepção dos veículos com lentes sujas que chegam a essa região limpa, o modelo não esteja adequado. Em resumo, mesmo que o veículo esteja em um domínio, o uso da aplicação sofre influência direta de características intrínsecas de clientes que trafegaram em outro domínio. Se muitos clientes com o mesmo histórico chegarem ao domínio, o servidor central pode considerar prudente requerer um novo treinamento.

Outro exemplo para ilustrar a influência do histórico no desvio de aprendizagem é voltado para a saúde. Aplicações utilizam dispositivos de IoT vestíveis para sensoriar parâmetros de saúde de pacientes, como frequência cardíaca, nível de oxigenação no sangue, glicemia, atividade do sono, entre outros. Através desses parâmetros, a aplicação pode recomendar ações para o usuário a fim de melhorar seu bem-estar ou até mesmo atuar diretamente com recomendações médicas e acionamento de socorro, caso perceba que o paciente necessita.

Nesse contexto, é importante que a aplicação avalie esses parâmetros com base no contexto em que o cliente esteja inserido a fim de analisar com fidelidade o seu estado de saúde. Por exemplo, em contextos de trabalho, é compreensível que um paciente esteja imerso em situações mais estressantes ou que durante a atividade física haja uma alteração momentânea em sua frequência cardíaca. Nesse sentido, a atividade ou contexto do paciente interfere diretamente na interpretação dos dados e parâmetros de sua saúde. Por exemplo, uma elevação do batimento cardíaco de um paciente é algo esperado durante a execução de uma atividade física, mas é um evento atípico durante o sono de um paciente. Dessa maneira, é possível que a aplicação julgue importante acionar o sistema de socorro para o paciente durante um contexto e para outro não, mesmo que os dados coletados pelos sensores sejam similares.

De forma similar, a detecção de elevação de um batimento cardíaco em diferentes usuários em uma mesma região física pode ser justificada por uma provocação do meio. Por exemplo, os usuários presentes em uma academia ou em uma quadra de esporte possivelmente estão com um média de batimento cardíaco maior do que usuários em suas casas. De forma análoga, uma elevação repentina da média dos batimentos cardíacos de usuários em uma situação de trabalho pode ser interpretada como uma situação de risco causada por estresse coletivo. Por outro lado, a mesma elevação repentina em uma sala de cinema, durante um filme de suspense, não deve apresentar os mesmos riscos para o usuário.

Ou seja, todo o contexto em que o usuário está inserido, incluindo o ambiente físico, os usuários, o histórico dos usuários e o meio em que a aplicação está inserida interfere diretamente na eficiência do modelo de AM.

Além disso, enfatiza-se que esse tipo de aplicação precisa ter uma alta acurácia. Em caso de falsos negativos, um paciente pode padecer de socorro em uma situação real de emergência, enquanto em casos de falsos positivos, um sistema de socorro com alto custo pode ser acionado de forma errônea e sem necessidade, acarretando prejuízo financeiros ou ocupando recursos que poderiam ser empregados em um sinistro verdadeiro. Assim, ao perceber que houve um desvio de aprendizagem, os servidores centrais devem imediatamente corrigir o aprendizado do seu modelo para garantir o cumprimento dos requisitos da aplicação.

Para viabilizar a atualização do modelo, o servidor central deve retreinar o modelo para novas lições, todavia durante esse processo há o problema do esquecimento catastrófico, esquecendo lições antigas e até então aprendidas. Considerando o dinamismo das aplicações,

com a mobilidade dos usuários, é possível que usuários que contribuíram para o aprendizado de um modelo reingressem ao domínio que porventura tenha migrado.

Durante a ausência desses usuários, o modelo pode ter sido treinado e esquecido das lições ensinadas por esses mesmos clientes, sendo necessário treinar o modelo novamente para atender as exigências dos clientes como já tinha ocorrido. Ou seja, há gastos computacionais, que muitas vezes são escassos, para ensinar novamente o modelo uma lição esquecida. À primeira vista, uma estratégia para amenizar esse problema é manter cópias dos modelos sempre que houver novos treinamentos, permitindo o chaveamento direto entre as cópias e atendendo a cenários históricos. Todavia, essa estratégia não é coerente pois os cenários não se repetem com exatidão, mas há apenas um retorno de um grupo de clientes a um domínio antigo. Em outras palavras, por mais que clientes antigos tenham retornado, o cenário em que eles encontram ao se conectar ao domínio já não é o mesmo. Portanto, ao restabelecer uma cópia antiga do modelo para atender aos clientes que retornaram, é possível que essa cópia prejudique os demais clientes.

Dessa forma, a estratégia mais eficiente é retreinar o modelo observando as lições anteriores. Segundo Martínez-Rego et al. [79], "este equilíbrio é conhecido como o dilema estabilidade-plasticidade, que é o contrapeso necessário entre a retenção de informações existentes e relevantes (estabilidade) e a aprendizagem de novos conhecimentos (plasticidade)". O aprendizado perfeito de novas lições manteria integralmente o aprendizado de tarefas antigas, sem mesmo necessitar que seja relembrado essas tarefas.

Todavia, esse aprendizado perfeito ainda não é factível devido a complexidade do modelo à medida que avançam no conhecimento. Além disso, o treinamento de modelos a partir de uma base de dados complexa pode acarretar um grande custo computacional inviabilizando o treinamento, principalmente quando há necessidade de atualização dos aprendizados em curto período. Em relação ao modelo de negócio algumas aplicações optam em manter a especialidade de modelos em ambientes distintos.

3.2 Aprendizagem Federada Personalizada e Multi-Tarefas

Segundo Dinh et al. [28], a heterogeneidade na distribuições de dados entre clientes dificulta a generalização do modelo para as diferentes bases de dados dos clientes, como destacado na Seção 2.1.3. Por outro lado, o treinamento a partir de um conjunto de dados amplo é fundamental para a eficiência do modelo na aplicação, visto que o treinamento local de um único cliente não é capaz de garantir a generalização do modelo devido à insuficiência de dados. Assim, o autor propôs o treinamento de modelos personalizados para cada cliente [28].

Em um contexto similar, Huang et al. [46] propuseram que cada cliente possua um modelo local personalizado. Para isso, o método FedAMP atribui a cada cliente o respectivo modelo personalizado com uma comunicação eficiente, definido a partir de uma combinação convexa de todas as mensagens que recebe. Assim, o método viabiliza a colaboração adaptativa subjacente entre pares de clientes.

Tanto Dinh et al. [28] quanto Huang et al. [46] alcançaram resultados relevantes na AF em contextos com heterogeneidade de dados, todavia os autores não consideraram a possibilidade de desvio de aprendizagem e heterogeneidade na disponibilidade de recursos computacionais dos clientes.

Em relação a solução de AMT, Sattler et al. [96] propuseram uma nova estrutura de Aprendizagem Federada Multi-tarefas (AFMT), destacando cenários com distribuições incongruentes de dados de clientes. Para isso, foi proposto uma ferramenta similaridade de cosseno que era usada para inferir se dois clientes têm distribuições diferentes ao gerar o dados. O resultado da inferência é usado para agrupar os clientes em *clusters*. A AFMT também foi estudada por [126], focando em cenários não-IID, não estacionários e explorando a correlação de modelos personalizados. Para isso, foi proposto uma estrutura de AFMT adaptativa, com uso de *cluster* na fase de treinamento. É importante destacar que o autor justificou a ocorrência do fenômeno de não-estacionariedade como uma possível consequência da mobilidade [126].

3.3 Soluções com Arquiteturas de AF

Diferentes trabalhos na literatura já analisaram o desafio de multidomínios em aplicações com AF com soluções baseadas em diferentes arquiteturas. Long et al. [69] propuseram um mecanismo de agregação multicêntrico com o objetivo de agrupar os clientes em vários *clusters*. Cada *cluster* possui seu próprio modelo global, que é definido a partir dos parâmetros dos modelos treinados localmente pelos clientes. Os *clusters* dessa proposta são formados a partir de um problema de otimização com o objetivo de minimizar a distância euclidiana L2 entre os gradientes locais para formar o modelo local.

Por sua vez, Briggs et al. [15] separam os clientes em *clusters* pela similaridade entre o resultado do treinamento local e o modelo global de cada cliente. Após a separação dos clientes, cada *cluster* treina um modelo de forma paralela e independente entre si. Segundo o autor, esta estratégia repercute em um treinamento de modelo que converge em menos rodadas de comunicação quando comparado a técnicas de treinamento sem *cluster* [15]. É importante destacar que o autor não considerou aspectos de abandono acarretados pela mobilidade ou de não-estacionariedade.

Outra solução similar foi proposta por Abad et al. [1] e denominada da AF hierárquico (HAF). Nessa solução, estações base de pequenas células orquestram os recursos para a AF. Periodicamente os modelos treinados em cada borda eram compartilhados pela nuvem entre todos os servidores para um consenso de um modelo global mais adequado para todos os usuários. A partir do instante em que o modelo é compartilhado a nuvem para que seja posteriormente atualizado na borda, os desafios referentes à mobilidade são amenizados, pois as bordas mantém a mesma instância de modelo de forma que o resultado do treinamento local pode ser considerado em qualquer borda, amenizando a migração de redes pelos clientes. Todavia, neste trabalho não há direcionamento que apontem para soluções para o desafio de multidomínios.

Apesar das vantagens mencionadas, Feng et al. [34] destacaram que é difícil avaliar em HAF o fator mobilidade na taxa de convergência da acurácia do modelo treinado. Além disso, Feng et al. [34] também destacaram que é difícil atribuir eficientemente clientes móveis, sob cenários de distribuição de dados não-IID, aos *clusters* que possuem servidores centrais com recursos limitados de comunicação e computação, devido aos custos adicionais

de comunicação e computação oriundos dessa estratégia. Todavia é preciso considerar aspectos sobre a heterogeneidade dos dados e a mobilidade dos usuários em aplicações de AF, cujo cenário é uma rede sem fio hierárquica. Assim, o autor analisou HAF com usuários móveis para propor um novo mecanismo de acesso, novas regras de treinamento local e estratégia de agregação de modelo, permitindo que usuários móveis participem do treinamento, aumentando a taxa de convergência e melhorando a precisão do modelo. Entretanto, o autor não avaliou o cenário considerando o princípio da não-estacionariedade e não observou a ocorrência de múltiplos domínios no problema da aplicação [34].

Uma proposta viável de uma arquitetura para as aplicações multidomínios, especialmente aquelas em que os domínios são definidos por regiões, é a computação de borda. Nessa arquitetura, dispositivos computacionais (denominados de servidores de borda) são instalados na borda da rede, disponibilizando recursos computacionais para a aplicação. Nessa arquitetura, o servidor central pode ser facilmente instalado em um desses servidores de borda, aproveitando os recursos computacionais desse dispositivo.

Nesse contexto, o EdgeFed é uma arquitetura para combinar as vantagens da AF e computação de borda. Nessa arquitetura, as saídas dos dispositivos móveis são agregadas no servidor de borda, diminuindo a frequência de comunicação global e melhorando a eficiência de aprendizado. Além da agregação, os servidores de borda também auxiliam os clientes no treinamento local, permitindo que os clientes se concentrem no treinamento de camadas baixas [118].

Nishio and Yonetani [89] também propuseram uma arquitetura de computação de borda móvel cujo servidor central era implantado no servidor de borda e coordenava o treinamento através de um algoritmo denominado FedCS. A primeira etapa desse algoritmo é coletar informações de um subconjunto de clientes aleatórios sobre a capacidade de computação e canais sem fio de comunicação. Com base nas informações coletadas, o algoritmo infere quais clientes desse subconjunto podem concluir o treinamento local em intervalo pré estabelecido, para finalmente selecionar apenas os clientes com essa capacidade, mitigando o atraso no treinamento [65, 89]

Também em uma arquitetura de computação de borda, Ganguly et al. [37] definiram o algoritmo CE-FL. No entanto, ao contrário das soluções anteriores, o servidor varia entre os ciclos de treinamento para lidar com a mudança na distribuição de dados e nas características

dos usuários à medida que o sistema evolui. Essa adaptação é crucial para mitigar o desvio de desempenho dos modelos, destacando a importância da aprendizagem contínua do modelo.

Outro trabalho relacionado a múltiplos servidores de borda e servidores centrais propôs o uso de clientes nas áreas de sobreposição de alcance de mais de uma borda para realizar treinamento local, calculando a média dos modelos recebidos dos servidores centrais que atuam nessas bordas. O resultado do treinamento local é enviado para todos os servidores que compartilharam inicialmente seus modelos. Assim, os clientes atuam como pontes, compartilhando conhecimento entre as bordas. Essa solução é interessante quando há restrições para compartilhar modelos entre bordas e se busca padronizar o domínio entre elas, pois à medida que o treinamento nesse formato se intensifica, os modelos dos diferentes servidores centrais tendem a se aproximar. Além disso, é relevante destacar que essa abordagem elimina a necessidade de uma nuvem para unificar o aprendizado do modelo, o que reduz o tempo geral de treinamento em comparação com abordagens convencionais da AF em nuvem [40].

3.4 Aprendizagem Federada e Desvio de Aprendizagem

O desvio de aprendizagem está relacionado ao desafio do treinamento em contextos com multidomínios. Nesse cenário, soluções centralizadas em um único modelo não são eficientes, pois os dados são heterogêneos e variam ao longo do tempo [50]. Segundo Jothimurugesan et al. [50], "quando diferentes clientes experimentam o desvio de dados em momentos diferentes, nenhum modelo global pode funcionar bem para todos os clientes. Da mesma forma, quando existem vários conceitos simultaneamente, nenhuma decisão de treinamento centralizada funciona bem para todos os clientes".

Casado et al. [20] propuseram um novo método, denominado de *Concept-Drift-Aware Federated Averaging*, que estende o FedAvg, aprimorando-o para adaptação contínua sob desvio de conceito. Neste método, os clientes enviam atualizações a qualquer momento, variando a taxa de participação entre eles. Dessa forma, o modelo global resultante é mais eficiente para os clientes que mais contribuíram no treinamento. O método se concentra na detecção e adaptação ao desvio de aprendizagem. No entanto, o método não avalia nem compreende como ou quem provocou o desvio. Para detectar o desvio, os clientes possuem duas memórias que armazenam dados de janelas temporais diferentes. Uma dessas memórias

é responsável por armazenar dados a curto prazo e auxilia o cliente na detecção do desvio. A segunda memória é a longo prazo e é usada para retreinar o modelo localmente. Como a memória a longo prazo mantêm um histórico de dados relevante do passado, além de dados recentes, o treinamento do modelo mantém o aprendizado de tarefas já aprendidas ao mesmo tempo que busca o aprendizado de novas tarefas [20]. Chow et al. [21] propuseram um novo método de agendamento de retreinamento de modelos para garantir a eficiência do treinamento federado em contextos dinâmicos ao mesmo tempo que reduz o tráfego de comunicação.

Tsiporkova et al. [101] propuseram uma metodologia para lidar com problemas de desvio de conceito em um ambiente de aprendizagem distribuída estatisticamente heterogêneo. Essa metodologia usa floresta aleatória para treinar diferentes modelos personalizados que são continuamente avaliados durante o uso. Os modelos são atribuídos a grupos de clientes semelhantes. O grau de semelhança entre os clientes é definido pela percepção do comportamento de desempenho do modelo global por cada cliente [101]. Ou seja, o cliente não adota estratégias mais complexas para avaliação de similaridade entre modelos locais e globais. Como a base de dados em ambientes não estacionários pode sofrer modificações, a percepção de um cliente sobre eficiência de um modelo também pode ser alterada, influenciando diretamente na definição dos grupos de clientes.

Kang et al. [51] reconheceram que o desvio de conceito é um desafio para AF, pois os clientes coletam os dados, que serão usados no treinamento do modelo, sob condições diferentes. Para enfrentar o desafio foi proposto melhorar as atualizações do modelo global. Assim o autor propôs empregar o método de Normalização de Peso para parametrizar novamente os pesos para ter média zero e variância unitária. Também é empregado o método de Normalização Adaptativa de Grupo para selecionar a média e o desvio padrão mais adequados para normalização de recursos com base no conjunto de dados [51].

O problema do desvio de aprendizagem também está diretamente ligado à necessidade de treinamento contínuo do modelo, cada vez mais comum em aplicações que precisam assimilar constantemente mudanças em seus domínios [37]. Essa demanda é similar ao comportamento humano, que está em constante aprendizado, sendo possível atualizar lições já aprendidas para melhorar o desempenho em uma tarefa, mesmo quando há mudança de conceitos. Apesar de a aprendizagem contínua viabilizar a assimilação de novas tarefas, é

importante que as lições já aprendidas não sejam esquecidas em detrimento da aprendizagem de novas lições. Caso isso ocorra, o aprendizado do modelo estaria limitado e não seria coerente afirmar que houve evolução do aprendizado. Esse problema é conhecido como esquecimento catastrófico, onde há uma substituição de especialização.

3.5 Considerações Finais

A fim de visualizar panoramicamente os trabalhos citados neste capítulo, sumariza-se as principais citações deste capítulo na Tabela 3.1. Nessa tabela, as citações são categorizadas a partir dos problemas e das soluções que foram abordadas. Assim, nas linhas são expostos os desafios, enquanto nas colunas estão as soluções. Com base nas discussões relativas aos artigos apresentados anteriormente, é possível identificar que para alguns trabalhos são abordados mais de um problema e a solução pode ser categorizada por mais de uma área. Por isso, algumas citações aparecem mais de uma vez na tabela. Antagonicamente, em alguns trabalhos são apresentadas de forma superficial melhorias para mais de um desafio, como a heterogeneidade de dados e do sistema, que foram amplamente citados. Nesse caso, considerou-se o foco principal do trabalho.

Observando a Tabela 3.1, identifica-se que não há na literatura, ao menos até onde essa revisão alcançou, estudos que apontem para contribuições na solução de todos os desafios simultaneamente. Todavia, como também já mostrado, diversas aplicações modernas estão imersas em contextos em que todos os desafios coexistem, tornando imprescindível o estudo de soluções que abordem simultaneamente esses desafios, em vez de propostas isoladas que trazem melhorias para uma área em detrimento de outras.

No caso do MoSimFeL o foco é na melhoria da seleção de clientes para aumentar a eficiência do treinamento do AF. Para isso, de forma inovadora aplica-se o conceito de similaridade na comparação entre modelos previamente treinados, a partir do histórico de cada cliente, com a perspectiva de inferir a contribuição de cada cliente em um novo treinamento antecipadamente. Isso permite que o servidor central selecione os clientes mais adequados para o progresso da aprendizagem do modelo e mitigar o efeito de desvio de aprendizagem.

Nesse contexto, destaca-se que até onde foi observado na revisão da literatura, a avaliação da similaridade na AF limita-se à análise de gradientes entre clientes no treinamento. No

Desafios \Soluções	**Multi-tarefas**	**Treinamento Personalizado**	**Similaridade**	**Arquitetura**
Multidomínios	–	–	[69, 96]	[69]
Abandono de clientes	–	–	[34]	[34]
Depreciação do modelo	[126]	[20, 101]	[126]	[37]
Mobilidade	[126]	–	[34, 126]	[34, 40, 89]
Heterogeneidade dos dados ou do sistema	[96, 126]	[28, 46]	[15, 34, 126]	[1, 15, 34, 37, 118]

Tabela 3.1: Categorização dos principais artigos abordados neste capítulo com base na proposta de solução e desafio.

entanto, a análise de similaridade entre clientes é inexequível se os clientes não participarem do mesmo ciclo de treinamento, ou seja, participando do treinamento da mesma instância de modelo. Além disso, a avaliação de similaridade concentrou-se em propostas direcionadas a definição de *clusters* com efeitos na etapa de agregação e na avaliação do gradiente dos clientes durante o próprio ciclo de treinamento. Isto significa que o cliente necessita disponibilizar recursos e executar o treinamento local e, somente após, a avaliação do gradiente é executada para analisar como a contribuição do cliente pode ser aproveitada na agregação do modelo global. Assim, o cliente empenha recursos e esforços computacionais sem a certeza de que será considerado no modelo global. Apesar das propostas avaliadas introduzirem melhorias no treinamento federado, a atuação na agregação não impede que clientes inadequados sejam escolhidos ou que recursos computacionais sejam alocados desnecessariamente.

Para o MoSimFeL, a seleção enviesada dos clientes também é baseada na mobilidade do cliente, inferindo sua capacidade de concluir o treinamento mediante a sua conexão com o servidor central. Dessa maneira mitiga-se os problemas relacionados ao abandono de clientes à medida que evita selecionar clientes que se ausentarão da rede antes de concluir o treinamento local. Por fim, para o MoSimFeL foi proposta uma arquitetura com múltiplos servidores centrais, preferencialmente instalados na borda da rede e que especializem o treinamento de um modelo considerando o domínio referente aos clientes que estão conectados a si.

Resumidamente, para o MoSimFeL buscou-se uma solução de aprendizagem federada

para contribuir para a solução dos desafios apresentados em aplicações com alto requisitos de eficiência de suas aplicações, adotando as estratégias:

1. treinar e manter pequenos modelos de AM exclusivos para cada servidor central para atender a domínios dos problemas;

2. padronizar um protocolo de identificação de desvio de aprendizagem para atualizar o treinamento do modelo a partir de novos treinamentos;

3. inferir indiretamente um conjunto de dados que favoreça o reforço de tarefas já aprendidas durante o retreinamento de um modelo, evitando o problema de esquecimento catastrófico, ao mesmo tempo que contemple os dados necessários para o aprendizado de novas lições;

4. inferir a contribuição em um novo treinamento de um cliente a partir da participação do histórico.

Capítulo 4

MoSimFeL

Neste capítulo, apresenta-se detalhadamente o MoSimFeL e uma discussão preliminar sobre o algoritmo. Para isso, inicialmente, na Seção 4.1, são descritas as restrições do cenário em que o algoritmo MoSimFeL é projetado. Na Seção 4.2, é apresentado o algoritmo FCKA, que é uma rotina para análise de similaridade com contexto de privacidade de dados e fundamental para o MoSimFeL. Na Seção 4.3, é apresentado o pseudocódigo do algoritmo com a explicação detalhada de sua implementação. Nas Subseções 4.3.1 e 4.3.2, são apresentados respectivamente uma discussão sobre a análise de similaridade no MoSimFeL e o efeito da mobilidade no algoritmo. Por fim, na Seção 4.4, retoma-se as principais considerações e conclusões deste capítulo.

Os símbolos que são usados na descrição são listados na Tabela 4.1. Referente a tabela, destaca-se o significado do símbolo M_b^a, adotado no restante desta tese. M_b^a refere-se a instância de um modelo de ML treinado, onde b é o identificador do servidor central que detém o modelo e coordenou o seu treinamento. Por sua vez, a é o identificador sequencial da instância de treinamento. Dessa maneira, a tupla (a, b) representa um identificador único de uma instância de modelo treinado. Por exemplo, M_1^2 é a segunda instância do modelo treinado pelo servidor central s_1 e, caso s_1 retreine o modelo, a nova instância será representada por M_1^3. Representa-se esse processo de treinamento do modelo por $M_1^2 \rightarrow M_1^3$.

Nesta tese também são utilizadas variações dessa nomenclatura sem explicitar os índices sobrescritos ou subscritos. Quando o índice b não for declarado, significa que trata-se de uma instância de modelo de um servidor qualquer. Por sua vez, quando o índice a não for declarado, significa que trata-se de uma instância de modelo de um servidor qualquer.

Tabela 4.1: Símbolos, Acrônicos e Descrição

S	Conjunto de servidores centrais
$\|\|S\|\|$	Número de servidores centrais
s_n	Um servidor central n, tal que $n \in N$
c_n	Um cliente c, tal que $n \in N$
M_b^a	Modelo de AM de instância a e treinado pelo servidor b, tal que $a, b \in \mathbb{N}$
$CKA(X, Y)$	Função *CKA* calcula a matriz de similaridade entre os modelos X e Y
$CKAN(X, Y)$	Função *CKAN* calcula um escalar de $CKA(X, Y)$
$M_b^a \rightarrow M_b^{(a+1)}$	Treinamento do modelo de AM do domínio do servidor central b atualizando a instância a para a instância $a + 1$
M_b	Instância qualquer de um modelo AM do domínio do servidor central b
M^a	Instância a do modelo AM do domínio de um servidor central qualquer
v_c	Velocidade do cliente c
a_s	Área de cobertura do servidor central s

4.1 Requisitos

No cenário ilustrado na Figura 3.1, aplicações de AF são implementadas em uma arquitetura de computação borda e servidores centrais são implementados na borda da rede criando domínios distintos e independentes entre si. Dessa forma, cada servidor central define seu próprio modelo global [74, 75].

Respeitando a privacidade de dados prevista na AF e contribuindo para dificultar a inferência indireta da base de dados dos clientes, os modelos globais de cada domínio não podem ser compartilhados entre servidores. Além disso, os clientes reservam o direito de não compartilhar suas bases de dados para aplicação. Como descrito na Seção 2.1, a limitação de acesso aos modelos aumenta a segurança da aplicação ao dificultar a inferência da base de dados dos clientes por agentes maliciosos. Adicionalmente, o impedimento ao compartilhamento dos modelos pode ser um requisito da aplicação, que opta por manter instâncias de modelos treinados distintos entre cada servidor central para preservar a especialização do modelo em cada domínio. Dessa maneira, um servidor central só tem acesso integral às

instâncias de modelos que coordenou o treinamento.

Estendendo o paradigma de computação de borda, uma estação central na nuvem se conecta a todos os servidores centrais, permitindo o compartilhamento de informações entre eles, desde que respeitando os requisitos de privacidade dos clientes. Esta estação central auxilia na coordenação da AF, especialmente na etapa de solicitação de informações sobre a mobilidade dos clientes e na mediação do processamento de similaridade utilizado na seleção dos clientes. A presença de um servidor na nuvem interligando os servidores centrais, como em uma topologia estrela, não é obrigatória no MoSimFeL e pode ser facilmente substituída por comunicação direta entre os servidores centrais através dos servidores de borda onde estão instalados. Sem perda de generalidade e consoante a Figura 3.1 assume-se a presença da estação central na descrição do cenário.

Informações dos clientes, como rotas de deslocamento, velocidade e recursos computacionais disponíveis, assim como a participação em treinamentos anteriores, podem ser compartilhadas na rede quando solicitadas pelos servidores centrais ou pela estação central. Ou seja, o acesso a informações dos clientes limita-se a dados usados como parâmetros da seleção. Todavia, destaca-se que um cliente só executa um treinamento para um servidor caso esteja sob o mesmo domínio.

Além disso, a estação central pode definir a qual domínio de servidor central um cliente se conecta com base em sua posição geográfica e histórico de conexões. Nesse sentido, o MoSimFeL não impõe especificações sobre os critérios que definem a conexão entre um cliente e um servidor central. No entanto, é necessário que haja previsibilidade nesse processo e que as informações para essa predição sejam plausíveis de serem compartilhadas sem infringir os requisitos de privacidade dos usuários dos clientes.

A imprevisibilidade da conexão entre clientes e servidores centrais limita a avaliação de mobilidade pelo MoSimFeL. No entanto, ainda é possível avaliar a similaridade, resultando no algoritmo SimFeL. O SimFeL é a versão do MoSimFeL focada exclusivamente na análise de similaridade. Da forma análoga, se a aplicação não disponibiliza o histórico de treinamento dos clientes, não é possível analisar a similaridade, mas ainda é possível avaliar a mobilidade, resultando na versão MoFeL.

Uma estrutura de grafo direcionado, $G(V, E)$, mapeia a relação de migração de clientes entre servidores centrais. No grafo G, V é o conjunto de vértices finitos que representam os

domínios. Ou seja, um vértice representa o conjunto formado por um servidor central, uma instância de modelo treinado de AF e um conjunto de clientes conectados a esse servidor. Por sua vez, E representa um conjunto finito de arestas definidas como (u, v), onde $u \in V$ e $v \in V$. Assim, uma aresta (v, u) indica um caminho para um cliente migrar do vértice v para o vértice u dentro do grafo.

No cenário descrito, cada cliente pertence a, no máximo, um único domínio em um determinado momento. Ou seja, apenas um vértice do grafo pode conter um cliente em um determinado momento, ou o cliente não está inserido em nenhum vértice.

Para ilustrar a migração de clientes entre servidores centrais, na Figura 4.1, ilustra-se um cenário de tráfego de veículos autônomos semelhante a uma micro cena do cenário da Figura 3.1. Nessa cena, os automóveis se conectam com os servidores de borda s_1, s_2 e s_3 ao se movimentarem e se aproximarem desses servidores.

Cada servidor de borda mantém instalado um servidor central da AF e os veículos são os clientes. Ao migrar de domínio, um cliente descarrega o modelo respectivo ao seu novo domínio, substituindo o modelo do domínio anterior [34].

O cenário ilustrado pela Figura 4.1 é representável por um grafo $G(V, E)$, conforme ilustrado na Figura 4.2.

Fonte: Adaptado de Macedo et al. [74]

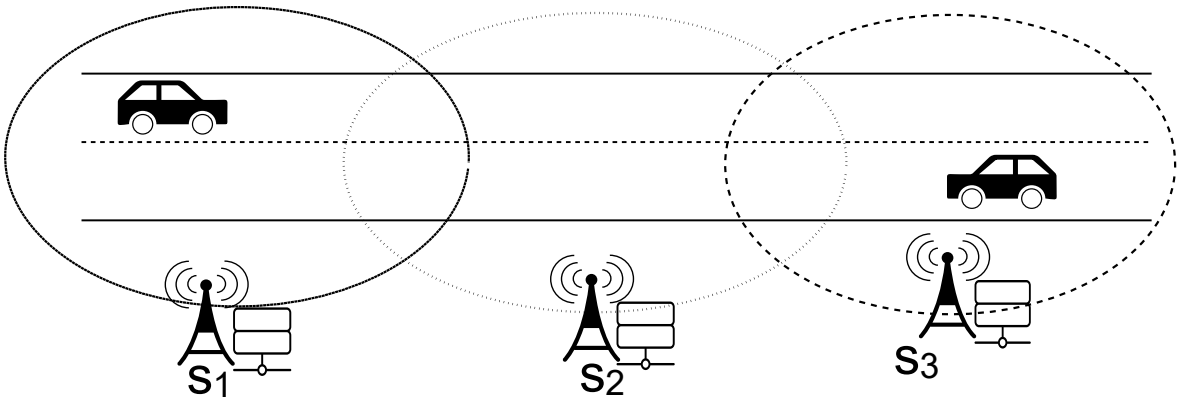


Figura 4.1: Exemplo de clientes migrando entre servidores centrais.

Ainda sobre o cenário da Figura 3.1, cada servidor mantém um modelo único e próprio devido à especialização do treinamento. Essa especialização é justificada pela regionalização dos dados usados e pela baixa similaridade entre os conjuntos de dados disponíveis nos

Fonte: Adaptado de Macedo et al. [74]

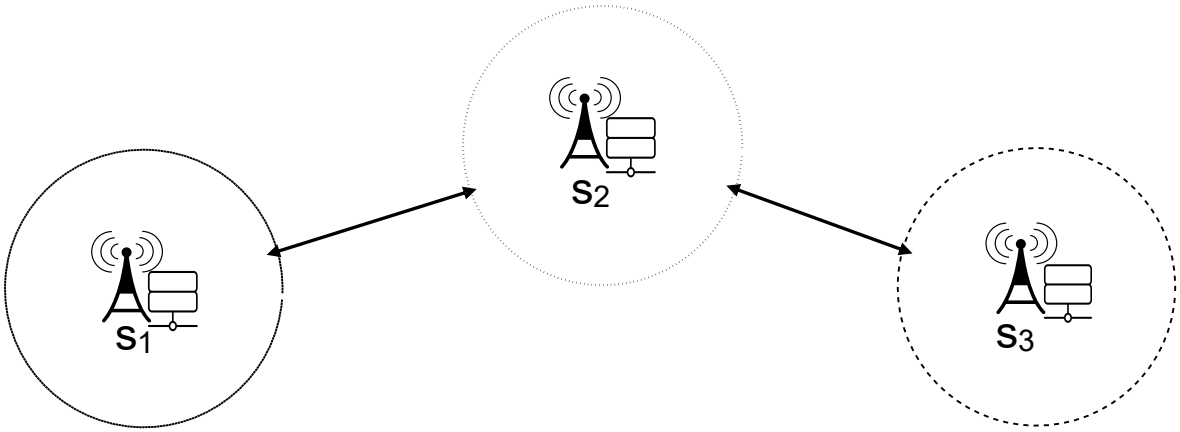


Figura 4.2: Exemplo de grafo referente a Figura 4.1.

domínios, ou pelo modelo de negócio que prioriza diferentes instâncias de modelos, considerando viável a monetização da aplicação.

Por fim, caso um cliente migre de domínio durante a execução de um treinamento local, ocorre o abandono do treinamento.

4.2 FCKA

O FCKA é um algoritmo de avaliação do CKA (descrito na Seção 2.2.1) que garante a privacidade dos dados dos clientes. Esse algoritmo é dividido em duas etapas: formação da base de dados D_{cka} (etapa *A*) e execução dos cálculos de similaridade (etapa *B*).

A análise de CKA, conforme descrito nas Equações 2.3 e 2.2, compara os modelos de AM a partir das matrizes de ativação (X e Y). Em outras palavras, os modelos são submetidos a uma entrada e definem suas respostas. Durante essa definição, cada camada terá uma saída que será usada pela camada subsequente até a definição da resposta final do modelo. As saídas intermediárias de cada camada, calculadas a partir de um conjunto de amostras, formam uma matriz de ativação para cada modelo.

Para tal, é necessário dispor de uma base de dados para produzir as matrizes de ativação. Essa base de dados deve representar o contexto ao qual os modelos serão submetidos. Idealmente, essa base de dados deve ser formada por amostras recém-adquiridas pelos clientes e que serão usadas pela aplicação. Todavia, a disponibilização de dados em AF representa um

grande desafio. Portanto, a questão central é: "*Como será construída a base de dados que será utilizada como parâmetro do CKA?*"

A construção da base de dados a partir da disponibilização de amostras coletadas pelos clientes não é necessariamente utópica no contexto de AF, dependendo dos requisitos de privacidade da aplicação, conforme exemplificado na Seção 2.1.3 [124, 127, 128]. Ganguly et al. [37] abordaram essa estratégia, incentivando os clientes que se voluntariarem na disponibilização de dados. O autor exemplificou isso através de uma aplicação veicular, onde dados oriundos de sensores dos automóveis e fotos de sinalizações não são necessariamente privados ou considerados sensíveis para todos os usuários. Dessa maneira, a aplicação oferece incentivos, como créditos de combustível, aos usuários que disponibilizarem esses dados. O compartilhamento seguro e voluntário desses dados pode ser assegurado por técnicas de criptografia que dificultem a interceptação, garantindo que apenas o servidor central tenha acesso a essas informações, conforme discutido nas Seções 2.1.4 e 2.1.3.

Todavia, algumas aplicações ou usuários podem considerar a disponibilização de dados reais uma violação de privacidade e, portanto, impedir a disponibilização de amostras. Assim, o FCKA apresenta um protocolo que define de forma distribuída o conjunto de dados para análise do CKA respeitando a privacidade. O protocolo FCKA é representado pelo digrama na Figura 4.3.

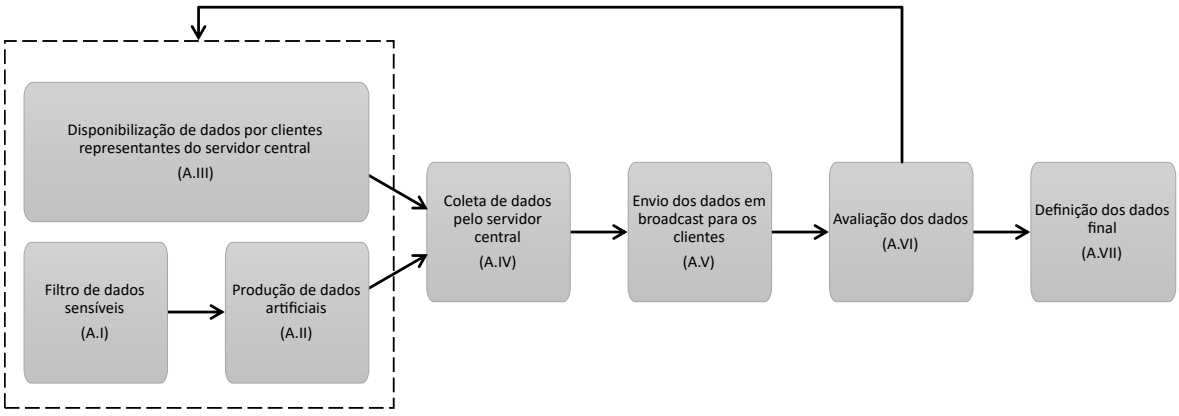


Figura 4.3: Formação de D_{cka} no FCKA (etapa *A*).

O FCKA inicia com a formação de uma base de dados para o servidor central (etapas *A.I*, *A.II*, e *A.III*, da Figura 4.3). Propõem-se duas estratégias para a formação dessa base de dados. A primeira é representada pelas etapas *A.I* e *A.II*, com os clientes fornecendo voluntariamente dados artificiais para o servidor central. Na etapa *A.I*, os clientes filtram

todos os dados com alguma informação sensivelmente privada, ou seja, que não podem ser compartilhados. Na etapa *A.II*, os dados que resistiram ao filtro anterior são usados como parâmetro para técnicas de fabricação de dados artificiais. Esses, e somente esses, dados artificiais são compartilhados com o servidor central. Apesar de serem artificiais, é importante que esses dados representem, mesmo que indiretamente, a base de dados à qual o modelo será submetido.

Nesse sentido, destaca-se que, para a avaliação de similaridade, a representatividade do conjunto de dados usado na avaliação com o conjunto de dados ao qual a aplicação será submetida é maleável. Portanto, o uso de dados reais é dispensável. Assim, um cliente pode definir um nível de representatividade dos dados compartilhados até o ponto em que permita o compartilhamento, reconhecendo que esses dados são suficientemente distintos dos seus dados originais.

Evidentemente, as etapas *A.I* e *A.II* demandam mais recursos computacionais dos clientes para a fabricação dos dados. Todavia, a demanda desses recursos pode ser mitigada com a adoção de técnicas eficientes e essa demanda pode torna-se desprezível em comparação com o esforço computacional demandando no treinamento local [87]. Além disso, é possível dividir as tarefas a fim de não sobrecarregar um cliente. Dessa forma, um cliente que fabrica dados artificiais para a avaliação do FCKA não será selecionado para o treinamento local, equilibrando o empenho de recursos computacionais no treinamento federado.

A segunda estratégia consiste na implementação de falsos usuários que façam a coleta de dados para o servidor central (etapa *A.III*). Esses usuários são comandados pelo servidor central e sua principal tarefa é monitorar o contexto em que os demais usuários estão inseridos. Dessa forma, esses usuários formam uma base de dados para análise da similaridade. Além disso, esses usuários não usufruem da aplicação e não podem participar das demais etapas do treinamento federado, pois a base de dados desses usuários não possui representatividade suficiente para ser usada no treinamento, limitando-se apenas à análise de similaridade.

A base de dados resultante das etapas *A.I*, *A.II* e *A.III* é compartilhada com o servidor central (etapa *A.IV*). Na etapa *A.V*, o servidor central distribui um subconjunto de amostras de sua base de dados para um conjunto de clientes, que, por sua vez, avaliam os dados recebidos e respondem se há um grau mínimo de representatividade dos dados (etapa *A.VI*). Em caso de avaliação positiva, o servidor central conclui o processo e usa essa base de dados para a

avaliação do CKA (etapa *A.VII*). Em caso de avaliação negativa, o servidor central retorna às etapas *A.I*, *A.II* e *A.III*, formando uma nova base de dados. Novamente, a definição sobre o grau de representatividade da base de dados na avaliação do CKA é uma atribuição da aplicação com base em seus requisitos.

Um servidor central s_1 pode requisitar a avaliação de similaridade do modelo de um outro servidor s_2 conforme o fluxo da Figura 4.4. O processo inicia com formação de D_{cka} por s_1, caso ainda não tenha definido essa base de dados (etapa *B.I*). Posteriormente, na etapa *B.II*, s_1 compartilha de D_{cka} com s_2. De posse da base de dados, cada um dos servidores define as matrizes de ativação (etapa *B.III*). Quando s_2 conclui a elaboração de sua matriz (Y), a compartilha com o servidor s_1, na etapa (etapa *B.IV*). Por fim, s_1 conclui o cálculo de CKA, com sua matriz X, etapa *B.V*.

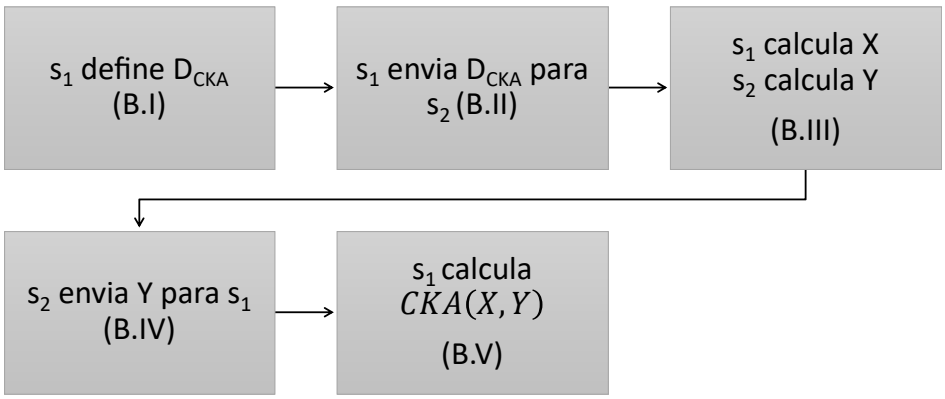


Figura 4.4: Execução do cálculo de similaridade (etapa B).

Em todo o processo do FCKA (incluindo as etapas *A* e *B*), não há o compartilhamento na rede de dados originais dos clientes, conservando o requisito de privacidade da AF. Até onde a revisão dos trabalhos nessa tese alcançou, não foi encontrado outro trabalho que propõe a construção de uma base de dados própria para cada servidor central, sendo um fato inovador. Além disso, o FCKA propõe que os clientes avaliem os dados. Assim, em vez do cliente compartilhar diretamente dados de sua base com o servidor central, o servidor pode verificar se sua base de dados é suficientemente adequada para a avaliação de similaridade. Por fim, destaca-se que as etapas *A* e *B* são independentes e podem ser flexibilizadas, dependendo dos requisitos de privacidade da aplicação e da disponibilidade de recursos computacionais para a execução do FCKA. Nesse sentido, é possível até mesmo eliminar uma das etapas ou modificá-las, a fim de mitigar a necessidade de recursos computacionais na execução.

4.3 MoSimFeL

O MoSimFeL é um algoritmo de coordenação de Aprendizagem Federada inspirado no FedAvg (descrito na Seção 2.1). Uma das diferenças entre esses algoritmos está na etapa de seleção de clientes. Enquanto no FedAvg a seleção é aleatória, o MoSimFeL enviesa a seleção a partir de dois critérios: a perspectiva de conclusão do treinamento mediante a mobilidade e a previsão da influência do cliente no treinamento. Esquematicamente, a seleção é representada pelo diagrama da Figura 4.5, através das etapas II e III.

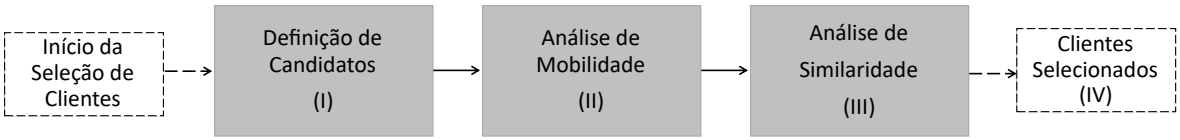


Figura 4.5: Representação de filtros na seleção dos clientes pelo MoSimFeL.

Esses algoritmos também se diferenciam na definição dos candidatos[1] para a etapa de seleção (etapa I, Figura 4.5). No FedAvg, um servidor central limita-se a selecionar clientes que estejam dentro de seu domínio, ou seja, todos os clientes do domínio são candidatos. Por sua vez, no MoSimFeL, o servidor central expande a avaliação de candidatos fora do seu domínio.

Assim, o MoSimFeL pode agendar a seleção de clientes para otimizar o treinamento, considerando a perspectiva de um futuro arranjo de clientes no domínio. Ou seja, o MoSimFeL avalia o melhor momento para iniciar o treinamento, considerando os clientes que estarão disponíveis no momento selecionado, devido à mobilidade dos candidatos que migram entre os domínios.

Para avaliar a disponibilidade dos clientes, o MoSimFeL considera a projeção da rota dos clientes. Nesse sentido, a rota é definida pela posição geográfica dos pontos em que o cliente trafega e pela previsão do momento em que passa por cada ponto. Considerando que a conexão de um cliente com um novo servidor central depende exclusivamente da posição geográfica do cliente e do servidor antecessor, a rota é suficiente para prever a migração de domínios durante a movimentação do cliente. Todavia, é possível estender a análise de disponibilidade para que sejam considerados aspectos além da mobilidade, tais como parâmetros

[1]Nesta tese, o termo “candidato” é usado para clientes que serão considerados na etapa de seleção

de QoS, confiança em um cliente ou disposição em empenhar recursos computacionais para o treinamento local.

A previsão da rota de um cliente não é necessariamente precisa, devido a erros de cálculo ou a eventos não determinísticos que podem ocorrer durante a movimentação do cliente, como a mudança inesperada de destino, acarretando em erros entre a previsão e a consolidação do trajeto. No entanto, o erro de previsão de rotas não inviabiliza o treinamento caso alguns clientes selecionados não estejam disponíveis em um domínio no momento previsto e agendado. Destaca-se que o MoSimFeL não é um algoritmo especializado em predição de rotas ou avaliação de rede. Espera-se que essas informações sejam fornecidas ao algoritmo como parâmetro a partir de rotinas especializadas nesses quesitos. O importante é que o MoSimFeL avalie essas rotas para inferir a disponibilidade do cliente em um domínio e direcione a seleção considerando essa informação.

Na linha 1 do algoritmo MoSimFeL, o servidor central define os candidatos a serem avaliados na etapa de seleção. Esse é um ponto crítico do algoritmo e está diretamente relacionado aos requisitos de eficiência da aplicação.

Se um servidor central define que os candidatos são os clientes que estão sob seu domínio, não há necessidade de avaliar a rota dos clientes de outros domínios. Esse caso requer menor esforço computacional para selecionar os clientes em comparação com uma candidatura que inclui clientes externos. Além da diminuição do número de clientes, a avaliação individual de um candidato interno [2] requer menor custo computacional do que a avaliação de um cliente externo. A avaliação de um cliente interno analisa somente o tempo restante para a próxima migração. Assim, o servidor central, ao selecionar um cliente, pondera se ele é capaz de concluir o treinamento antes de se desconectar. Para um cliente externo, a avaliação também deve calcular quando ele se conectará ao servidor.

A definição dos candidatos também interfere na etapa de avaliação da similaridade. Em um caso específico, se o último treinamento de todos os candidatos for a última instância do modelo do servidor central, a avaliação de similaridade limita-se à comparação do gradiente resultante do treinamento local de cada candidato no último treinamento do modelo do

[2]Nesta tese, o adjetivo "interno" é utilizado para se referir a clientes ou candidatos que estão no domínio de um servidor central durante a seleção de clientes para um novo treinamento federado. De forma análoga, o adjetivo "externo" é utilizado para clientes ou candidatos que não estão conectados a esse servidor.

Algorithm 2 Algoritmo MoSimFeL

Require: t_i, instante inicial de disponibilidade; t_f, instante final de disponibilidade; γ_i, instante final de disponibilidade; γ_f, instante final de disponibilidade; ι, instante final de disponibilidade; η, numero de clientes para treinamento;

1: $C \leftarrow definir\_candidatos$
2: $\forall c \in C, rota_c \leftarrow solicitar\_sota(c)$
3: $\forall c \in C, (I_c, F_c) \leftarrow prever\_chegada\_saida(rota_c, t_i, t_f)$
4: $C_D \leftarrow filtro\_mobidade_1 \| filtro\_mobidade_2$
5: **for** $c \in C_D$ **do**
6: $\quad servidor_c, instancia_c, gradiente_c \leftarrow solicitar\_historico\_treinamento(c)$
7: $\quad$ **if** $(instancia_c, instancia_s)$ não foi avaliado **then**
8: $\quad\quad fcka_{instancia_c, instancia_s} \leftarrow calcular_fcka(servidor_c, instancia_s)$
9: $\quad$ **end if**
10: $\quad \gamma_c \leftarrow \alpha * fd_c(\beta * gradiente_c + \omega * fcka_c)$
11: $\quad$ **if** $\gamma_i < \gamma_c < \gamma_f$ **then**
12: $\quad\quad selecionados \leftarrow selecionados + c$
13: $\quad$ **end if**
14: **end for**
15: **procedure** $filtro\_mobidade_1(\forall c \in C, (I_c, F_c))$
16: $\quad M_C \leftarrow \sum_{c \in C} \frac{I_c}{|C|}$
17: $\quad \forall c \in C, L_c \leftarrow |M_C - I_c|$
18: $\quad \forall c \in C, fd_c \leftarrow \frac{L_c - min_{c \in C}(L_c)}{max_{c \in C}(L_c) - min_{c \in C}(L_c)}$
19: $\quad maiores\_fd_c \leftarrow ordernar_{c \in C}(fd_c)$
20: $\quad C_D \leftarrow maiores\_fd_c[: num\_candidatos\_disposniveis]$
21: $\quad$ **return** C_D
22: **end procedure**
23: **procedure** $filtro\_mobidade_2(\forall c \in C, (I_c, F_c))$
24: $\quad \forall c \in C, D_c \leftarrow F_c - G_c$
25: $\quad \forall c \in C, C_D \leftarrow C_D + c \, if D_c \geq 0$
26: $\quad$ **return** C_D
27: **end procedure**

servidor. Nesse contexto, não há necessidade de avaliar a similaridade entre os modelos históricos dos candidatos, visto que todos possuem o mesmo histórico imediato, repercutindo na diminuição do esforço computacional.

A diminuição do conjunto de candidatos pode impactar significativamente a otimização do treinamento devido à redução das opções disponíveis para seleção. Isso ocorre porque, ao restringir o conjunto de candidatos, existe a possibilidade de excluir clientes cujas bases de dados poderiam contribuir de maneira positiva para a eficiência do treinamento. Em um cenário extremo, essa restrição pode tornar inviável a obtenção de uma solução eficiente para a seleção dos melhores candidatos. Portanto, é fundamental que a aplicação encontre um equilíbrio entre o esforço computacional e a precisão na escolha dos clientes ideais.

De forma análoga, incorporar os clientes de todos os domínios ao conjunto de candidatos acarreta um alto esforço computacional na fase de seleção de clientes. Se o número de clientes aumentar exacerbadamente, a solução computacional pode se tornar inviável ou demandar tanto tempo para processamento que cause um descompasso entre a tomada de decisão e a movimentação do cliente. Ou seja, enquanto a seleção é processada, os clientes já podem ter mudado de posição ou rota.

Nesta tese, a definição do melhor conjunto de candidatos não será abordada de forma aprofundada, e serão adotadas propostas arbitrárias para a solução, como a candidatura de clientes geograficamente posicionados dentro de um raio pré-estabelecido como parâmetro do MoSimFeL. Todavia, é possível propor a avaliação de estratégias mais eficientes, como a alteração desse raio durante o ciclo da aplicação a partir da avaliação dinâmica de sua influência no treinamento. Outra estratégia é a definição de um índice de confiabilidade de rotas para priorizar a candidatura de clientes com previsibilidade mais assertiva. Assim, cada aplicação deve estipular regras para a definição do conjunto de candidatos.

Na sequência, o MoSimFeL avalia a disponibilidade dos clientes, considerando a mobilidade e a estimativa de tempo gasto para a conclusão do treinamento local. Ou seja, o MoSimFeL filtra o conjunto inicial de candidatos considerando a sua disponibilidade em um intervalo de tempo, através das linhas [2-4], formando um novo subconjunto de clientes (C_D) e agendando o instante previsto em que o arranjo de clientes nesse conjunto se consolidará.

No instante agendado, o simulador central avalia a precisão de sua previsão com base nos clientes que estão efetivamente conectados a ele. Se a quantidade de clientes confirmados

for adequada, o algoritmo progride para uma nova filtragem. Caso contrário, o servidor pode aguardar um intervalo de tempo para compensar atrasos de alguns clientes ou iniciar um novo agendamento, voltando à avaliação de disponibilidade. Além disso, o servidor central pode complementar o conjunto C_D com clientes presentes para aumentar o número de opções para a próxima etapa.

No segundo filtro, os clientes pertencentes a C_D são analisados e é atribuído a cada um desses clientes um fator de contribuição, descrito nas linhas [5-12]. Somente após a análise do fator de contribuição, é selecionado o conjunto final de clientes para iniciar o treinamento local.

Dessa maneira, o algoritmo MoSimFeL filtra o conjunto de clientes candidatos em duas etapas distintas e sequenciais, definindo a seleção a partir da interseção dos dois filtros. Considerando a independência dos critérios de avaliação (disponibilidade e fator de contribuição), nas próximas seções, cada critério será detalhadamente explicado.

Assim, nas próximas subseções, os dois filtros serão detalhadamente descritos. Na Subseção 4.3, descreve-se a avaliação do critério de mobilidade e, na Subseção 4.3.2, descreve a avaliação indireta da base de dados clientes a partir da análise de similaridade.

4.3.1 O MoSimFeL e a Mobilidade

Após a definição dos candidatos, inicia-se o processo de requisição de informações de todos os clientes candidatos sobre sua mobilidade. Para cada cliente, é solicitada sua rota (linha 1), considerando pontos geográficos e o instante previsto para o comparecimento em cada ponto. O MoSimFeL não especifica qual dispositivo (servidor central, cliente ou outro dispositivo) realiza a predição dessas rotas. Nesta tese, sem perda de generalidade, foi arbitrariamente definido que o cliente prevê sua própria rota e a informa aos servidores centrais que solicitarem.

Na linha 2, para cada rota solicitada ao cliente (c), o MoSimFeL prevê o próximo instante em que o cliente ingressará em seu domínio (I_c), no intervalo de tempo $[t_i, t_f]$. Se um cliente não tem perspectiva de entrar no domínio do servidor central requisitante, ele é descartado da seleção. Nesse sentido, o MoSimFeL não define que o cálculo de I_c é responsabilidade do servidor central. Nesta tese, adotou-se o servidor como responsável dessa tarefa, mas ela também pode ser atribuída ao cliente. Ou seja, o cliente definiria o I_c e informa ao servidor

quando for questionado. Ainda na linha 2, o MoSimFeL prevê o instante em que o cliente c se desconectará do domínio F_c. A avaliação de F_c é importante para analisar se o cliente consegue concluir o treinamento antes de se desconectar do servidor central.

A vantagem de atribuir a responsabilidade da definição de I_c ao cliente é aumentar sua privacidade, pois não seria necessário fornecer sua rota ao servidor central, mesmo que fosse usada exclusivamente na definição de I_c. Por outro lado, a desvantagem é atribuir mais tarefas ao cliente, que empenharia recursos computacionais no cálculo de I_c.

Claramente, se o conjunto inicial C, definido na linha 1, conter apenas os clientes conectados ao domínio do servidor central, o valor de I_c é desconsiderado e a avaliação de disponibilidade do cliente considera apenas a sua perspectiva de permanecer no domínio. Nesse sentido, na linha 4 propõe-se duas rotinas para avaliação da disponibilidade do cliente $filtro\_mobidade_1$ e $filtro\_mobidade_2$.

Na rotina $filtro\_mobidade_1$, considera-se a seleção de clientes externos ao domínio. Assim, na linha 16, o servidor central calcula a média de todos os I_c (M_C) e define-se, para cada cliente, a distância de I_c com M_C (linha 17). Essa distância representa o quanto um cliente estaria distante entre sua perspectiva de chegada ao domínio e a média entre os demais clientes. Essa distância é denominada de L_c (linha 18).

Na linha 19, os valores de L_c são normalizados para a definição de fd_c. Dessa forma, $fd_c \in [0, 1]$ e representa um fator normalizado, de forma que quanto mais próximo a 0, mais próximo I_c é de M_C.

Na linha 20, os valores de fd_c são ordenados para que apenas os clientes com menor fd_c sejam incluídos no conjunto C_D, tornando-se aptos para a próxima etapa da seleção. Assim, os clientes com I_c mais próximos à M_C são priorizados e a quantidade dos clientes que passam pela próxima fase é um parâmetro do algoritmo.

O objetivo é encontrar, de forma empírica, o maior número possível de clientes em um intervalo de tempo reduzido. A abordagem empírica reconhece que as previsões de rotas e, consequentemente, de I_c, possuem uma margem de erro. Portanto, executar cálculos para definir este intervalo com precisão não é viável, dado que os parâmetros não são precisos. Nessa estratégia, não são considerados parâmetros como o tempo restante previsto para a permanência no domínio do servidor central requisitante do treinamento, nem o tempo estimado para a execução do treinamento local. Isso pode resultar em altas taxas de abandono

de treinamento, pois não se distingue entre um cliente prestes a sair do domínio e outro com previsão de permanência prolongada. A solução proposta considera apenas o instante de chegada do cliente ao domínio do servidor requisitante do treinamento.

No entanto, nessa proposta, espera-se que o tempo médio de execução de um treinamento local entre os clientes seja significativamente menor do que o tempo médio de permanência no domínio, garantindo que uma quantidade significativa de clientes conclua o treinamento quando requisitado. Portanto, embora o abandono de treinamento ainda possa ocorrer na solução apresentada, espera-se que seja menor em comparação com uma solução que não considera sequer o instante de entrada dos clientes no domínio.

Na rotina $filtro\_mobidade_2$, considera-se apenas os clientes internos ao domínio. Assim, na linha 24, para cada cliente, estima-se a capacidade do cliente em concluir o treinamento, considerando a sua perspectiva de permanência F_c no domínio e o tempo estimado para conclui o treinamento G_c. Consecutivamente, na linha 25, apenas os clientes com capacidade de conclui o treinamento passam pelo filtro local, ou seja, $se(F_c - D_c) \geq 0$.

Novamente, destaca-se que o objetivo principal do MoSimFeL não é inferir a rota dos clientes ou prever a capacidade de um cliente concluir o treinamento. Logo, as rotinas $filtro\_mobidade_1$ e $filtro\_mobidade_2$ podem ser substituídas por novas rotinas que considerem com maior precisão a movimentação dos clientes mediante um modelo de mobilidade mais próximo ao contexto da aplicação.

4.3.2 O MoSimFeL e o CKA

O segundo filtro (etapa III, Figura 4.5) do MoSimFeL ocorre entre as linhas [5-12] e somente os clientes que passaram pelo filtro do etapa II são avaliados. Calculas-se a etapa III somente no instante agendado da etapa II, considerando o arranjo de clientes que efetivamente se conectaram ao servidor. Se a consolidação desse arranjo for muito discrepante da projeção inicial, é possível reiniciar o processo com um novo agendamento na etapa II. Esse intervalo na seleção dos clientes, calculando as etapas I e II em momentos distintos, é justificado pelas falhas de previsibilidade nas rotas. Se a previsão das rotas for precisa, as etapas I e II podem ser calculadas sequencialmente e sem interrupções.

Na linha 5, é solicitado a cada cliente informações sobre sua última participação em um treinamento, incluindo o servidor central que coordenou esse treinamento ($servidor_c$), o

modelo resultante ($instancia_c$), e sua contribuição ($gradiente_c$).

Ainda para cada cliente, calcula-se o FCKA entre os modelos $instancia_c$ e $instancia_s$, sendo $instancia_s$ o atual modelo de requisição do cliente. A análise de FCKA é armazenada na linha 8. Em seguida, na linha 10, um fator de contribuição (γ_c) para o treinamento para cada cliente é calculado, considerando o gradiente, a similaridade e sua disponibilidade no servidor central. Esse fator é definido como $\gamma_c = \alpha \cdot fd_c(\beta \cdot gradiente_c + \omega \cdot fcka_c)$.

Sobre γ_c é importante destacar que os índices fd_c, $gradiente_c$ e $fdcka_c$ são ponderados respectivamente por α, β e ω. Esses coeficientes equilibram a importância dos fatores na definição de γ_c. Nesta tese foi adotado a definição arbitrária desses coeficientes. Todavia, é possível adotar outras estratégias para definição desses valores considerando os requisitos da aplicação. Sem perda de generalidade, os valores de α, β e ω podem ser dinâmicos, sendo alterados durante a seleção para enviesar a escolha de clientes por diferentes critérios.

Por exemplo, se uma aplicação priorizar a extinção de abandono de treinamentos, é aumentar o coeficiente α em relação ao demais para valorizar a avaliação de disponibilidade do cliente no ciclo de treinamento. Se uma aplicação priorizar a acurácia resultante do treinamento, os coeficientes β e ω deve ser superestimado em relação ao α.

Na linha 11, os clientes são filtrados pela sua estimativa de contribuição γ_c de forma que os clientes com γ_c fora do limite $[\gamma_i, \gamma_f]$ são desconsiderados da seleção. Os limites γ_i e γ_f também são parâmetros da aplicação. Na linha 11, também é possível selecionar os clientes considerando uma ordenação de todos os resultados γ_c, priorizando os clientes com maiores ou menores índices.

Por fim, na linha 12, os candidatos que passaram pelo filtro são aleatoriamente selecionados. Ao todo são selecionados $\eta \cdot \iota$ clientes, onde η é o número de clientes que o servidor central almeje no treinamento e ι é um coeficiente que representa uma margem de segurança, tal que $\iota \geq 1$.

O número de clientes selecionados acima do almejado é justificado pelos erros na avaliação de disponibilidade dos clientes, ou seja, de I_c. Além disso, essa margem de reserva na seleção de clientes pode considerar outras falhas no treinamento, como a perda de comunicação, falta de bateria ou indisponibilidade de recursos computacionais para o treinamento local devido ao surgimento de outras demandas do usuário.

Nesse sentido, é importante que ι garanta uma margem segura para assegurar de que

ao menos η clientes selecionados executem o treinamento de forma adequada. No entanto, ι não deve ser excessivamente grande, pois um número muito maior que η de clientes no treinamento pode causar problemas, conforme descrito na Seção 2.1.

Destaca-se que, durante a solicitação do histórico de treinamento dos clientes na linha 5, considera-se apenas a última participação de treinamento de um cliente. No entanto, é possível estender a análise do MoSimFeL para considerar todo o histórico de treinamento do cliente, incluindo suas diversas participações em treinamentos de modelos em diferentes domínios. Essa extensão permite uma análise detalhada sobre a base de dados do cliente sem precisar acessar os dados de forma direta.

4.4 Considerações Finais

Durante a descrição do MoSimFeL, foram discutidas possíveis melhorias e modificações do algoritmo. Destaca-se que o MoSimFeL permite alterações em suas fases, o que pode resultar em melhorias dependendo dos requisitos das aplicações e cenários, mas também em uma diminuição de eficiência em outros. Nesse contexto, em trabalhos futuros pode-se avaliar com precisão a influência das modificações propostas para contribuir com a evolução do treinamento federado.

Os conceitos implementados para o MoSimFeL podem ser empregados no treinamento tradicional. Assumindo que há um modelo treinado (M_s), que precisa ser treinado novamente a partir de uma nova base de dados, e que o custo do treinamento é proporcional ao tamanho da base de dados, é possível separar a base de dados em conjuntos menores e distintos (X) e treinar diferentes modelos para cada um dos conjuntos (M_x). Assim, com os modelos M_x, calcula-se a similaridade entre M_x e M_s e avalia-se como cada base de dados X pode contribuir distintamente para o treinamento de M_s. Ou seja, é possível avaliar antecipadamente o efeito de uma base de dados no treinamento de um modelo qualquer. Com isso, é possível definir qual base de dados será usada no treinamento, reduzindo o custo do treinamento. Claramente, essa análise precisa ser fundamentada e comprovada, sendo mais uma proposta de trabalho futuro.

Por fim, destaca-se que a proposta do MoSimFeL atende aos requisitos de privacidade de dados estipulados pela AF. Nesse sentido, em todas as etapas em que houve necessidade do

uso de dados, o acesso aos dados foi controlado para não infringir esses requisitos, conforme descrito na Seção 4.2.

Capítulo 5

O simulador FLSimulator

Neste capítulo é apresentado o FLSimulator. O FLSimulator é um programa de simulação de algoritmos de coordenação de AF para simular cenários com mobilidade, não-estacionários e com múltiplos domínios. Seu principal objetivo é avaliar a eficiência do treinamento a partir de diferentes métricas que serão descritas na Seção 5.3. O simulador também possibilita avaliar os algoritmos de coordenação com um único servidor central, em cenários estacionários e sem mobilidade.

A arquitetura do FLSimulator foi projetada para viabilizar a extensão do simulador a partir do desenvolvimento de novas estruturas. Para isso, o simulador disponibiliza um arcabouço de código computacional, com estruturas abstratas e interfaces para implementação. Isso permite que pesquisadores possam implementar novos algoritmos de coordenação de AF, novas rotinas de controle de mobilidade ou diversos cenários.

A simulação de cenários com mobilidade, múltiplos domínios, não-estacionários e a capacidade de extensão torna o FLSimulator customizável contemplando variados desafios de simulação. Até onde foi pesquisado, esse é o primeiro simulador de AF com essas características.

Nas próximas seções, descreve-se o FLSimulator com detalhes: na Seção 5.1, apresenta-se a arquitetura de desenvolvimento do simulador; na Seção 5.2, descreve-se o ciclo de vida da simulação; na Seção 5.3, explica-se as métricas que a atual versão do FLSimulator é capaz de analisar e; na Seção 5.4, aborda-se de forma detalhada a interação do FLSimulator com o simulador SUMO; Por fim, na Seção 5.5, retoma-se os principais aspectos deste capítulo, concluindo a apresentação do FLSimulator e destacando trabalhos futuros para a expansão

do simulador.

5.1 Arquitetura do FLSimulator

O FLSimulator é formado por um conjunto de módulos que se relacionam entre si a partir de rotinas específicas para contribuírem com a simulação. Cada módulo pode ser expandido a partir da implementação das classes ou interfaces que o rege.

Para um pesquisador, a implementação de novas estruturas para o FLSimulator é mais rápida do que a implementação completa de um novo simulador, pois durante a extensão, rotinas essenciais e complexas para o ciclo de vida da simulação e intrínsecas a cada módulo já estão desenvolvidas. Dessa forma, o FLSimulator é uma solução atrativa para pesquisadores permitindo a customização do simulador de forma fácil e rápida.

Na Figura 5.1 é apresentado os módulos e a interação entre eles. O FLSimulator é estruturado pelos módulos: FLSMobility, FLSNet, FLSDistribuitor, FLSTimer, FLSCore, FLAnalyser e FLSCreator. Os módulos serão explicados detalhadamente. Todavia, enfatiza-se que a explicação se restringe a versão atual do FLSimulator e que os módulos podem ser estendidos viabilizando as mais diversas modificações.

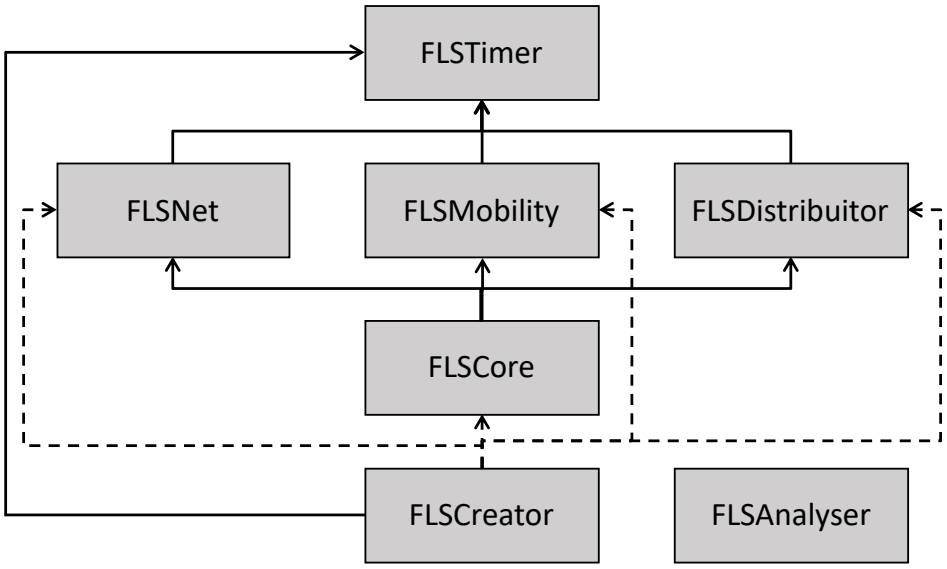


Figura 5.1: Esquema representando a divisão e interação de módulos do FLSimulator.

FLSCore

O FLSCore controla a estrutura de dados principal do simulador. Essa estrutura de dados é baseada em um grafo direcionado $G(V, A)$, conforme descrito na 4.1. Nesse contexto, o FLSimulator restringe que cada cliente só integre a um domínio por vez.

No FLSCore há a implementação das entidades servidor central e cliente. Cada cliente possui uma memória que durante a simulação, a depender da estratégia de distribuição de dados, acumula dados para o treinamento local. Em relação ao servidor central, é importante destacar que durante a simulação, as entidades que representam os servidores mantêm o histórico de treinamento de seus modelos, a instância corrente do modelo e o histórico de conexões de clientes no domínio que coordenam. Além disso, nessas entidades há um arcabouço de código com atributos e rotinas fundamentais para a execução do simulador.

Apesar da disponibilização desse arcabouço, caso o desenvolvimento de uma simulação requeira novos atributos e funcionalidades, é possível implementar novas classes estendendo as já existentes. Por exemplo, na atual versão do simulador, apenas clientes possuem rotinas e atributos relacionados à mobilidade, considerando que os servidores centrais não mudam de posição geográfica durante a simulação.

As entidades clientes e servidores centrais não compartilham dados e informações que infrinjam os requisitos tradicionais do treinamento federado, como observado no algoritmo FedAvg [80]. Por exemplo: um cliente não pode compartilhar dados com outro cliente; um servidor central não pode acessar a base de dados de um cliente e; um servidor central não pode se comunicar com clientes de outros domínios.

Para viabilizar a maleabilidade de restrições e requisitos das aplicações, o FLSCore dispõe de uma entidade denominada de Mundo. Essa entidade garante o acesso a todos os clientes e servidores centrais e viabiliza a customização de novas soluções de treinamento a partir da atenuação de restrições ao modelo tradicional proposto para o FedAvg. Por exemplo, na proposta do MoSimFeL, um servidor central precisa de informações sobre a mobilidade de usuários, mesmo que não estejam em seu domínio, e do histórico de gradientes resultantes do treinamento local desses usuários em outros domínios. Nesse contexto, considerando o desenvolvimento do simulador, o acesso a essas informações são feitas a partir da entidade Mundo. Outro exemplo, é a solução prevista em [15] em que os clientes de forma distribuída acessam informações dos seus gradientes a fim de agregarem modelos globais intermediá-

rios. A nível de programação, como as entidades clientes não se comunicam diretamente com outros clientes, é possível que haja essa disponibilização a partir da entidade Mundo. O desenvolvedor também pode optar por não usar essa entidade e expandir a implementação das entidades cliente e servidor central como já mencionado.

Através da Figura 5.1 observa-se que todos os demais módulos do simulador interagem com FLSCore, recebendo-o como parâmetro. Isso é justificado pois todos os demais módulos acessam informações do cenário (a partir de clientes e servidores centrais) para suas rotinas ou alteram o cenário simulado. Por exemplo, o módulo FLSMobility é responsável pela mobilidade dos dispositivos simulados e atualiza, a cada passo da simulação, a posição geográfica do cliente.

Por sua vez, o módulo FLSDistribuitor é responsável por auxiliar na formação da base de dados de cada cliente. Para isso, o FLSDistribuitor acessa informações sobre o posicionamento do cliente para decidir qual amostra de dados o cliente receberá. Dessa maneira, o FLSDistribuitor lê e altera as entidades do FLSCore.

FLSCreator

O módulo FLSCreator define a criação do cenário. Nesse sentido, o FLSimulator permite a simulação de cenários dinâmicos, incluindo a adição ou remoção de clientes, servidores centrais e até mesmo alteração da estrutura do grafo, no FLSCore, durante a execução da simulação. O FLSCreator é o primeiro módulo a ser chamado na simulação, lendo os parâmetros definidos pelo usuário. Este módulo gera a estrutura do grafo do FLSCore e cria instâncias para os clientes e servidores centrais conforme a esses parâmetros. Além disso, a implementação do FLSCreator é diretamente relacionada aos demais módulos, contribuindo para a configuração específica e necessária de cada módulo. Por isso, na Figura 5.1, o FLSCreator é representado como a origem dos outros módulos.

FLSMobility

O módulo FLSMobility é responsável por gerenciar a mobilidade dos clientes no cenário simulado e recebe como parâmetro o FLSCore. Atualmente esse módulo dispõe de duas implementações para gerenciar a mobilidade dos cenários. A primeira implementação faz

integração com o simulador *Simulation of Urban Mobility* (SUMO). O SUMO é software de simulação de tráfego de código aberto, projetado para lidar com grandes redes. O SUMO simulação permite a simulação de pedestres e diferentes veículos, integrando-se a um conjunto de ferramentas para criação de cenários [70]. Ou seja, o FLSimulator interage com outro simulador de mobilidade através de uma API disponibilizada pelo SUMO e, dessa forma, garante de forma terceirizada uma simulação robusta e próxima à cenários reais de mobilidade.

A segunda implementação do FLSMobolity simula a mobilidade através de uma série de distribuições estatísticas e estocásticas que definem quando um cliente muda de domínio a partir das restrições de mobilidade. A análise estatística para essa mobilidade considera parâmetros como velocidade e percurso a ser percorrido por um cliente ao se conectar a um novo domínio. Dessa maneira, as futuras conexões de um cliente são inferidas de forma indireta a partir da mudança de sua posição.

Apesar de ser baseado em distribuições estatísticas, essa implementação não é eficiente para cenários de mobilidade urbana com trânsito de automóveis, ônibus, pessoas e outros tipos de veículos, pois esses cenários requerem uma simulação mais complexa. Todavia, essa implementação é adequada para avaliação de cenários mais abrangentes e focados no modelo de negócio. Por exemplo, essa implementação possibilita simular a perspectiva de um cliente em mudar de domínio a partir do interesse da aplicação, definidos na camada de negócio, e de modelos estocásticos que emulam o comportamento do cliente.

O FLSMobility também interage e recebe como parâmetro o FLSCore, modificando parâmetros do cliente relacionados a seu posicionamento. Como já mencionado, a atual versão do módulo FLSCore não contempla a mobilidade de outros dispositivos que não sejam os clientes do cenário e, por isso, apenas a entidade cliente armazena dados relacionados à mobilidade, tais como velocidade de movimentação, ponto de partida, rota planejada e ponto de chegada. Todavia, o usuário do simulador também pode implementar sua própria estratégia de mobilidade, expandindo a implementação do FLSCore, do FLSMobility e de outros módulos caso necessário. Nesse contexto, a expansão do módulo FLSMobility, assim como os demais, pode fazer uma integração com outras ferramentas de simulação, tal como a integração com o SUMO.

O módulo de mobilidade pode ser diretamente ligado ao módulo FLSCreator, pois o mó-

dulo FLSCreator não apenas configura a estrutura interna do FLSCore como também auxilia na definição de parâmetros para o FLSMobility. Por exemplo, na implementação com o SUMO, o FLSCreator define antecipadamente os arquivos de configuração que serão usados na execução do SUMO, garantindo que a definição do cenário simulado e representado na estrutura de dados do FLSCore seja compatível com o cenário simulado no SUMO. Assim, para cenários em que a mobilidade é regida pelo SUMO, há uma implementação específica do FLSCreator.

FLSNet

O módulo FLSNet define e controla atribuições da rede conectando os clientes aos servidores centrais, considerando tanto parâmetros da aplicação quanto da camada física do protocolo de comunicação. Nesse módulo, é definido como e a qual domínio os clientes se conectam à medida que movimentam durante a simulação. Para isso, o FLSNet consome informações do FLSCore a cada passo da simulação para identificar o estado corrente de posicionamento e conexões entres clientes e servidores centrais para a tomada de decisão quanto a mudança de conexões e parâmetros relacionados ao QoS.

A atual versão do FLSNet dispõe de uma implementação simples para definir as conexões entre clientes e servidores centrais. Nessa implementação, para cada servidor central é definido uma área de abrangência cujo centro é a posição do servidor. Os clientes priorizam a conexão com o servidor central que está mais próximo de si e que a área contemple a sua posição. Essa implementação contempla a análise de algoritmos como o FedAvg e o MoSimFeL, propostos neste trabalho, visto que ambos não avaliam aspectos de QoS na coordenação do treinamento federado.

Assim como os demais módulos, novas implementações podem ser desenvolvidas para o FLSNet. Nesse contexto, é possível integrar esse módulo com outros programas de simulação de redes com o intuito de garantir estruturas de simulação confiáveis e consolidadas na literatura acadêmica, atendendo a soluções de algoritmos que precisem desse parâmetro. Sobre isso, propostas futuras para ampliação do simulador, visam a integração com as ferramentas OMNET++ e NS_3 [43, 103]. O OMENT++ e o NS_3 são simuladores de redes baseados em eventos discretos. Através desses simuladores e a extensão do módulo FLSNet, estações e antenas podem ser distribuídas no cenário simulado do SUMO. Essas estações

simulam servidores de borda que implementam servidores centrais e gerenciam redes sob o protocolo 5G, por exemplo [98]. Dessa maneira, o requisito para definir a conexão entre cliente e servidor central traspassaria a simples aproximação entre dispositivos para requisitos de QoS, definido pelo servidor de borda e protocolo 5G.

FLSDistribuitor

O módulo FLSDistribuitor distribui os dados para os clientes durante o ciclo de vida da simulação. Esses dados são armazenados pelos clientes e usados para o treinamento local. Na versão atual do simulador, a implementação desse módulo define que o fluxo de distribuição é regido por distribuições estatísticas que analisa quando e quantas amostras cada cliente recebe, considerando seu posicionamento, domínio e o instante de simulação.

Dessa maneira, o FLSDistribuitor analisa os atributos dos clientes e servidores, como tempo de vida e posição dos dispositivos, para distribuir dados a cada instante de simulação. A atual versão implementada também viabiliza a distribuição de amostras exclusivas para os servidores centrais para que sejam usadas na avaliação dos modelos treinados a depender do algoritmo de coordenação do treinamento federado. Esse módulo também é responsável por gerir a não-estacionariedade do sistema, definindo arbitrariamente ou por distribuição estatística eventos de mudança de contexto para os servidores centrais e clientes.

O FLSDistributor é relacionado à base de dados que alimenta os dados usados na aplicação. Na versão atual, há implementações para gerenciar a base de dados MNIST (Modified National Institute of Standards and Technology) [59]. Essa base de dados contém amostras de imagens digitalizadas de dígitos e letras manuscritas, permitindo o desenvolvimento e a simulação de aplicações de reconhecimento de imagens. O FLSDistributor também dispõe de um arcabouço que facilita a expansão do módulo, permitindo a incorporação de novas estratégias de distribuição e outras bases de dados.

FLSTimer

O módulo FLSTimer controla a execução da simulação e a cada instante coordena a execução de rotinas dos outros módulos para que possam atualizar o cenário simulado, conforme as respectivas atribuições, progredindo o ciclo de vida da simulação. Por exemplo, FLSTimer

usa o FLSMobility para atualizar a posição dos dispositivos do cenário. Por isso, conforme a Figura 5.1, o FLSTimer recebe como parâmetro os módulos FLSDistribuitor, FLSCore, FLSNet e FLSMobility.

FLSAnalyser

O último módulo apresentado na Figura 5.1 é o FLSAnalyser. Esse módulo arquiva informações sobre a simulação, como o registro de mudanças de domínio pelos clientes, a participação em treinamentos, o empenho de recursos computacionais e falhas no treinamento local devido a mudanças de domínio. Além disso, esse módulo estrutura essas informações para facilitar a análise, permitindo a plotagem de gráficos e o agrupamento de dados da mesma categoria para a definição de métricas.

Na Figura 5.1, o FLSAnalyser é apresentado isolado dos demais módulos, sem conexões, pois ele não os recebe como parâmetros. Em vez disso, o FLSAnalyser, no âmbito do código de programação, é usado em qualquer trecho do projeto para acessar informações relevantes para a avaliação da simulação. Isso é possível porque esse módulo não interfere no ciclo de vida da simulação, ao contrário dos demais módulos.

O FLSAnalyser não pode ser sobrescrito. Todavia, caso o usuário da aplicação queira novas métricas, é possível implementar novas rotinas sem conexão com as rotinas do FLSAnalyser e usá-las para coletar novas informações.

A principal vantagem da customização dos módulos do FLSimulator é permitir a extensão do simulador para contemplar diversas estratégias de treinamento federado. Todavia, é fundamental que durante a implementação de novos módulos, o desenvolvedor restrinja o acesso aos dados conforme as definições da aplicação e do cenário. Por exemplo, no MoSimFeL, qualquer servidor central pode acessar informações sobre a rota dos clientes, independentemente de estarem no mesmo domínio. Por outro lado, a implementação do FedAvg é mais simples e não necessita de acesso a essas informações de quaisquer clientes. Em ambos os casos, cabe ao desenvolvedor ajustar as restrições ao acesso das informações conforme necessário.

Por fim, a implementação atual dos módulos é baseada em técnicas de treinamento síncrono, como as técnicas FedAvg e MoSimFeL. Portanto, a agregação do modelo global é executada apenas com os resultados do treinamento local de todos os clientes. Além disso,

o ciclo de vida dos módulos é concorrente, mas os treinamentos individuais dos clientes são executados sequencialmente, iniciando o treinamento de um cliente apenas após o término do anterior. Destaca-se que, em contextos reais, o treinamento federado é executado de forma distribuída e paralela. No entanto, a estrutura dos módulos permite o paralelismo da simulação ao criar fluxos de processos durante a implementação das interfaces e ao distribuir o treinamento dos clientes em diferentes dispositivos.

5.2 Ciclo de Vida da Simulação

O ciclo de vida da simulação é ilustrado no fluxograma da Figura 5.2. Neste fluxograma, o ciclo começa através da inicialização do módulo FLSAnalyser (passo 1). Logo após, o módulo FLSCreator assume o fluxo da simulação e cria um cenário inicial para a simulação contendo clientes, servidores centrais e posicionando geograficamente os dispositivos no cenário criado (passos 2 e 3). A instância desse módulo permanece ativa durante toda a simulação e pode criar clientes, servidores centrais, assim como eliminar os já existentes, à medida que a simulação é executada.

Na versão atual do simulador, é possível distribuir os dispositivos no cenário através de distribuições aleatórias, regulares ou atribuir a responsabilidade da alocação inicial ao módulo de mobilidade. A distribuição aleatória, define coordenadas aleatórias entre limites previamente definidos para cada dispositivo enquanto a opção uniforme distribui os dispositivos em espaços regulares e iguais entre si.

No passo 4, o módulo FLSMobility assume a simulação e define os parâmetros de mobilidade para cada dispositivo criado no passo anterior.

Após a criação inicial do cenário, o módulo FLSDistrbuitor é iniciado, carregando a base de dados que será usada para distribuir dados entre os clientes (passo 5) durante o ciclo da simulação. Nesse módulo também é possível tratar os dados previamente, com aplicação de filtros e fabricação artificial de novos dados antes que sejam passados para os clientes.

No passo 6, o módulo FLSTimer assume a simulação. A cada passo da simulação, o FLSTimer solicita que os demais módulos façam alterações no cenário simulado, alterando a posição dos dispositivos, distribuindo dados ou controlando a remoção e criação de novos dispositivos.

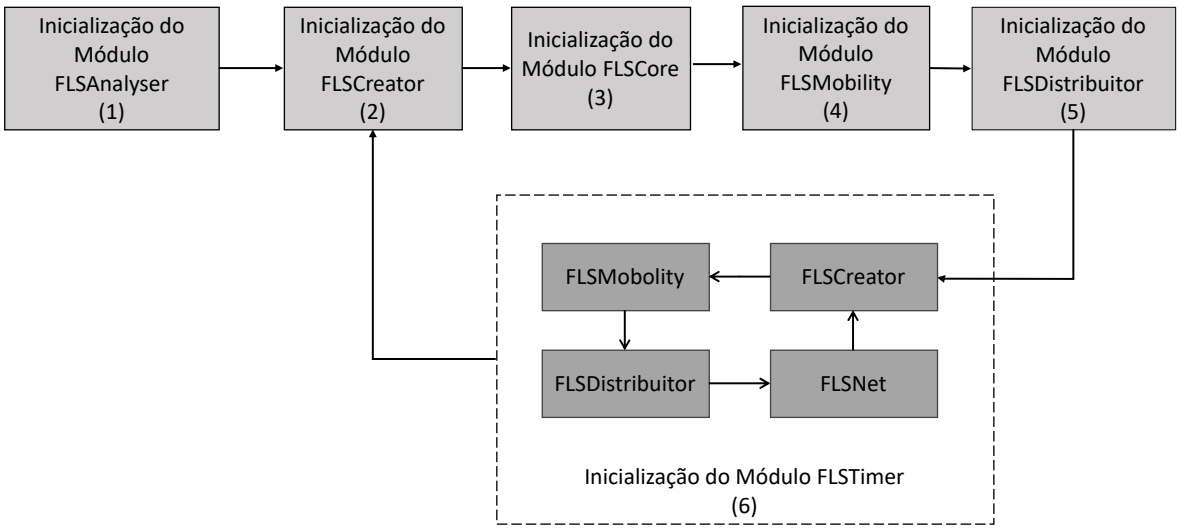


Figura 5.2: Ciclo de vida de uma simulação no FLSimulator.

O FLSTimer controla a simulação durante todo o intervalo definido. A execução desse intervalo estabelece um ciclo de simulação. Durante esse intervalo, os treinamentos federados são executados a partir da requisição dos servidores centrais e considerando os requisitos da aplicação, que define quando um modelo precisa ser submetido a um novo treinamento.

Com o término desse ciclo, a simulação volta à criação de um novo cenário através do módulo FLSCreator para que o ciclo de simulação se inicie novamente a partir de um novo cenário. Esse recomeço é importante devido à natureza não determinística da simulação. Dessa maneira, o usuário pode programar a repetição da simulação para que os resultados convirjam e não representem alguma situação específica e probabilisticamente raro. Nesse sentido, o único módulo que não é reiniciado é o FLSAnalyser pois esse módulo coleta os resultados de todos os ciclos e é capaz de disponibilizar a média das métricas entre todos os ciclos.

5.3 Métricas do FLSAnalyser

O FLSAnalyser dispõe de um conjunto de rotinas para arquivar e processar informações durante os ciclos de simulação para a avaliação de métricas. As informações armazenadas são listadas a seguir:

1. ACC_N: Acurácia média do modelo treinado M^x a partir da avaliação da base de

testes apenas dos clientes que estão no domínio imediatamente após a conclusão do treinamento e que não participaram do treinamento da instância anterior do modelo M^{x-1}.

2. ACC_O: Acurácia média do modelo treinado M^x a partir da avaliação da base de testes apenas dos clientes que estão no domínio imediatamente após a conclusão do treinamento e que participaram do treinamento da instância anterior do modelo M^{x-1}.

3. ACC_A: Acurácia média do modelo treinado M^x a partir da avaliação da base de testes apenas dos clientes que estão no domínio imediatamente após a conclusão do treinamento.

4. $ACC(t)$: Acurácia média do modelo treinado M^x a partir da avaliação da base de testes dos clientes que estão no domínio no instante t.

5. QT: Número de ciclos de treinamento executados.

6. QA: Número de clientes que abandonaram o treinamento.

7. QT: Quantidade de ciclos de treinamento executados durante a simulação.

As métricas listadas descrevem a performance do algoritmo de coordenação e devem ser avaliadas conjuntamente na análise do cumprimento de requisitos da aplicação por esse algoritmo.

A métrica ACC_N descreve se o treinamento consegue evoluir o aprendizado do modelo para atender aos requisitos da aplicação para os clientes novos da rede, ou seja, $ACC_N > \epsilon$. De forma análoga, ACC_O descreve se após o treinamento, em busca de aprender tarefas para atender aos novos clientes, o modelo mantém a qualidade do aprendizado para os clientes antigos, ou seja $ACC_O > \epsilon$. Caso $ACC_O < \epsilon$ é um indício que o retreinamento do modelo está acarretando o problema do esquecimento catastrófico.

Por sua vez, ACC_A e $ACC(t)$ expressam a capacidade do algoritmo em atender os requisitos da aplicação para todos os clientes independente do momento em que se conectam a um servidor central. No melhor dos casos, o algoritmo de coordenação consegue manter a regra $\forall t \in [t_i, t_f], ACC(t) > \epsilon$.

Em relação a QR, a métrica avalia a eficiência do treinamento do algoritmo de coordenação. O valor de QR está diretamente relacionado a alocação de recursos computacionais para o treinamento do modelo. Quanto mais ciclos de treinamento, mais recursos computacionais serão empenhados para o treinamento local e para agregação do modelo global, além de ocupar o canal de comunicação. A métrica QR é direcionamento relacionada a métrica QMC, que define a quantidade média de treinamentos locais solicitados pelo número de clientes e expressa o volume de recursos computacionais alocados. Portanto, quanto maior QMC, o algoritmo de coordenação é menos efetivo.

Nesse sentido, a seleção de clientes é fundamental e o fator decisivo para a performance de treinamento dos algoritmos. Uma seleção eficiente de clientes aumenta a performance do treinamento, acelerando a convergência do aprendizado e diminuindo a quantidade de ciclos de treinamento. Obviamente, o número de ciclos deve ser analisado junto com as métricas ACC_N, ACC_O e ACC_A para que considere se os ciclos de treinamento conseguiram atender aos requisitos da aplicação.

A métrica QA avalia o empenho de recursos computacionais dos clientes na execução incompleta de treinamentos locais. Ao iniciar um treinamento local e alocar recursos computacionais para isso, o cliente almeja que o resultado de seu esforço computacional seja considerado na agregação do modelo global.

É importante enfatizar que as métricas definidas nessa seção e propostas pelo simulador restringem a definição requisitos da aplicação pela acurácia. Todavia, é possível estender a análise para avaliação de outras métricas.

5.4 FLSimulator e SUMO

Para o FLSMobility, duas técnicas de mobilidade foram implementadas. A primeira analisa o grafo da estrutura de dados e dispara eventos estocásticos para cada cliente, para executar sua transição de domínio considerando as restrições de arestas do grafo.

A segunda é uma implementação que interage com o simulador SUMO. Devido à complexidade dessa implementação, esta subseção descreverá com detalhes a implementação desse simulador de mobilidade.

O FLSMobility interage com SUMO através de duas APIs: Traci e Libsumo . Traci

é acrônimo do termo em inglês *Traffic Control Interface* que, traduzido para o português, significa Interface de Controle de Tráfego. Essa API viabiliza o acesso ao controle da simulação de tráfego que está sendo executada, permitindo acessar o status da simulação e os parâmetros dos objetos simulados. Além disso, a API permite o controle da simulação [70]. Todavia, o Traci usa comunicação por soquete e pode apresentar uma sobrecarga de comunicação. Por sua vez, a Libsumo é uma biblioteca que viabiliza o acesso à simulação de forma mais eficiente, sem a necessidade de comunicação por soquete [70].

Para executar a simulação por meio dessa implementação do módulo FLSMobility, recomenda-se usar ferramentas complementares do SUMO para preparar o ambiente de simulação.

Inicialmente a ferramenta OSMWebWizard[1] é recomendada para simular cenários de mobilidade a partir de mapas reais. Segundo a documentação oficial do SUMO, OSMWebWizard permite a seleção de um trecho do mapa do Openstreetmap. O Openstreetmap é um banco de dados de mapas construído colaborativamente e voluntariamente. A elaboração do mapa por viagens de carro, fotos, vídeos, rastreamentos de GPS e por outras informações geoespaciais [90]. e configuração de demanda de tráfego aleatório. Essa ferramenta exporta arquivos auxiliares no formato XML para configurar a simulação do SUMO.

Com base nesses arquivos é possível ter acesso antecipado aos veículos que trafegam durante a simulação e uma previsão de suas rotas, considerando o ponto de partida, o ponto de destino e o caminho para cada agente no tráfego, como veículos, humanos e embarcações marítimas. Dessa maneira, as informações sobre a previsão de rotas utilizadas pelo MoSimFeL são implementadas pelo próprio SUMO. Na Figura 5.3, para exemplificar a operação do OSMWebWizard, é possível visualizar a tela de seleção do mapa focando a região do município de Campina Grande, no estado da Paraíba, no Brasil.

Após a exportação do cenário inicial pelo OSMWebWizard, é possível editar configurações no mapa gerado a partir da ferramenta Netedit[2] (também disponibilizada pelo SUMO). Essa ferramenta permite alterar rotas, acrescentar semáforos, definir o sentido de vias e diversas outras edições do mapa. Além disso, é possível criar um mapa completo com sua

[1]Disponível em: https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html - Acesso em: 13 de Maio de 2024.

[2]Disponível em: https://sumo.dlr.de/docs/Netedit/index.html - Acesso em: 13 de Maio de 2024.

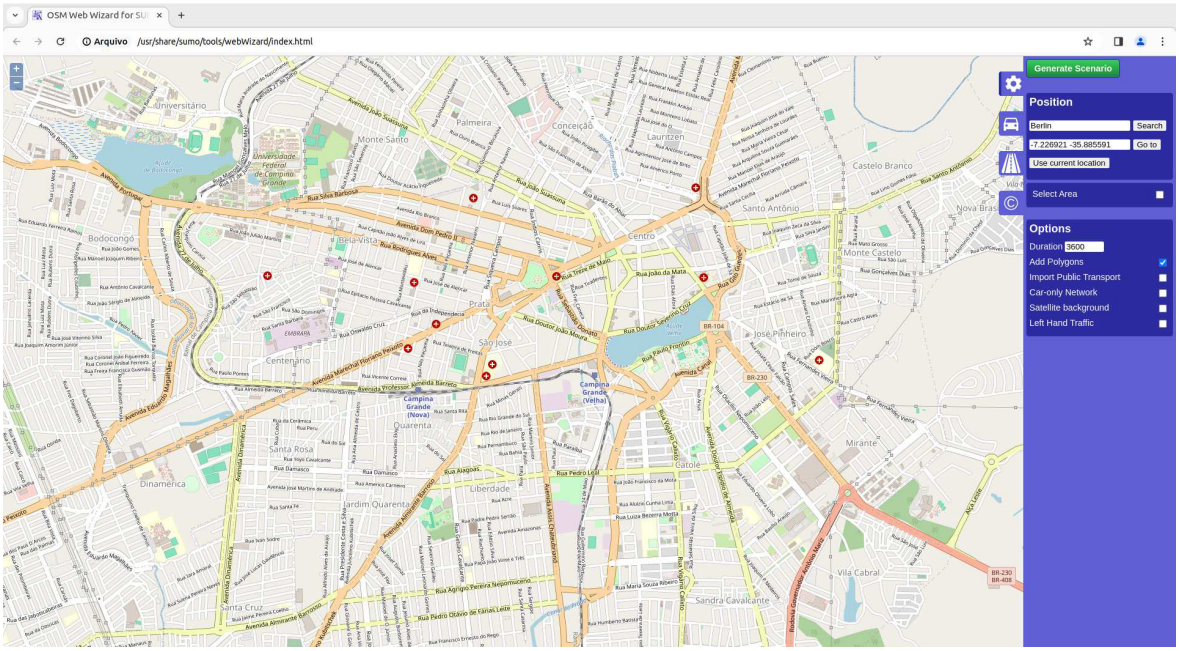


Figura 5.3: Mapa da cidade de Campina Grande do aplicativo OSMWebWizard.

interface gráfica, sem a necessidade do OSMWebWizard. Netedit é um editor visual de rede. Esse editor permite criar redes ou editar redes existentes, a partir de uma interface gráfica [70].

As viagens dos veículos podem ser previamente definidas pela ferramenta RandomTrips[3]. Segundo a documentação do SUMO, essa ferramenta gera um conjunto de viagens aleatórias, definindo os pontos de origem e destino de cada agente de tráfego de forma a partir de uma seleção aleatória e uniforme ou com uma seleção aleatória a partir de suas opções de distribuições estatísticas [70].

Outra ferramenta que pode auxiliar na implementação do cenário de simulação é a Duarouter[4], que gera os caminhos de cada viagem, considerando que os agentes de tráfego tenham uma origem e um destino já definidos. Nesse processo é definido o trajeto que os agentes percorreram e a previsão de saída de alguns pontos durante sua trajetória. O duarouter calcula rotas de veículos considerando o caminho mais curto. Essas rotas podem ser usadas como parâmetro pelo SUMO [70].

Dessa maneira, as informações sobre a previsão de rotas utilizadas pelo MoSimFeL são implementadas pelo próprio SUMO. Nesse contexto, é importante que essa previsão de rota,

[3]Disponível em: https://sumo.dlr.de/docs/Tools/Trip.html - Acesso em: 13 de Maio de 2024

[4]Disponível em: https://sumo.dlr.de/docs/Netedit/index.html - Acesso em: 13 de Maio de 2024.

assim como os instantes citados, são previsões do simulador e podem sofrer oscilação durante a execução da simulação, mediante o efeito de eventos não determinísticos simulados pelo SUMO. Assim, a partir dessa implementação, a simulação trabalha sob rotas imprecisas e o algoritmo de coordenação de AF que utilize os parâmetros da rota dos clientes deve considerar essas incertezas.

As ferramentas citadas exportam o resultado de suas rotinas em arquivos no formato XML, que podem ser lidos pelo SUMO para parametrizar a simulação de mobilidade. Além disso, todos esses arquivos podem ser gerados automaticamente pelo FLSCreator. Todavia, as ferramentas disponibilizadas pelo SUMO possuem interface amigável que, de forma interativa, facilita a criação de cenários a serem simulados.

Assim, é possível ignorar o uso das ferramentas propostas e desenvolver as rotinas respeitando a semântica do XML de entrada do SUMO e os parâmetros da simulação de forma autônoma. Porém, a implementação atual do módulo do FLSimulator optou em receber esses parâmetros para gerenciar a mobilidade e, por isso, deve-se gerar esses parâmetros externamente ao FLSimulator. Outra possibilidade, é criar uma implementação da interface FLSMobility para gerar esses parâmetros.

Por fim, a execução da simulação pode ser graficamente visualizada através da ferramenta *sumo-gui*, exemplificado na captura de tela da Figura 5.4[5], em que é possível controlar o tempo de simulação para acompanhar graficamente a simulação, conforme ilustrado pela Figura [70].

5.5 Considerações Finais

O desenvolvimento do FLSimulator atende à demanda de simulações de algoritmos de coordenação de treinamento federado no cenário proposto. Até onde esta pesquisa alcançou, não há outra ferramenta de simulação com as mesmas características do FLSimulator ou que atenda à demanda de experimentos desta tese, justificando o seu desenvolvimento.

Destaca-se que a arquitetura do simulador permite sua fácil expansão, viabilizando seu uso por outros pesquisadores ao otimizar o desenvolvimento de novas funcionalidades. Todavia, reconhece-se que o simulador está em suas primeiras versões e, por isso, requer atuali-

[5]Disponível em: https://sumo.dlr.de/docs/sumo-gui.html - Acesso em: 13 de Maio de 2024.

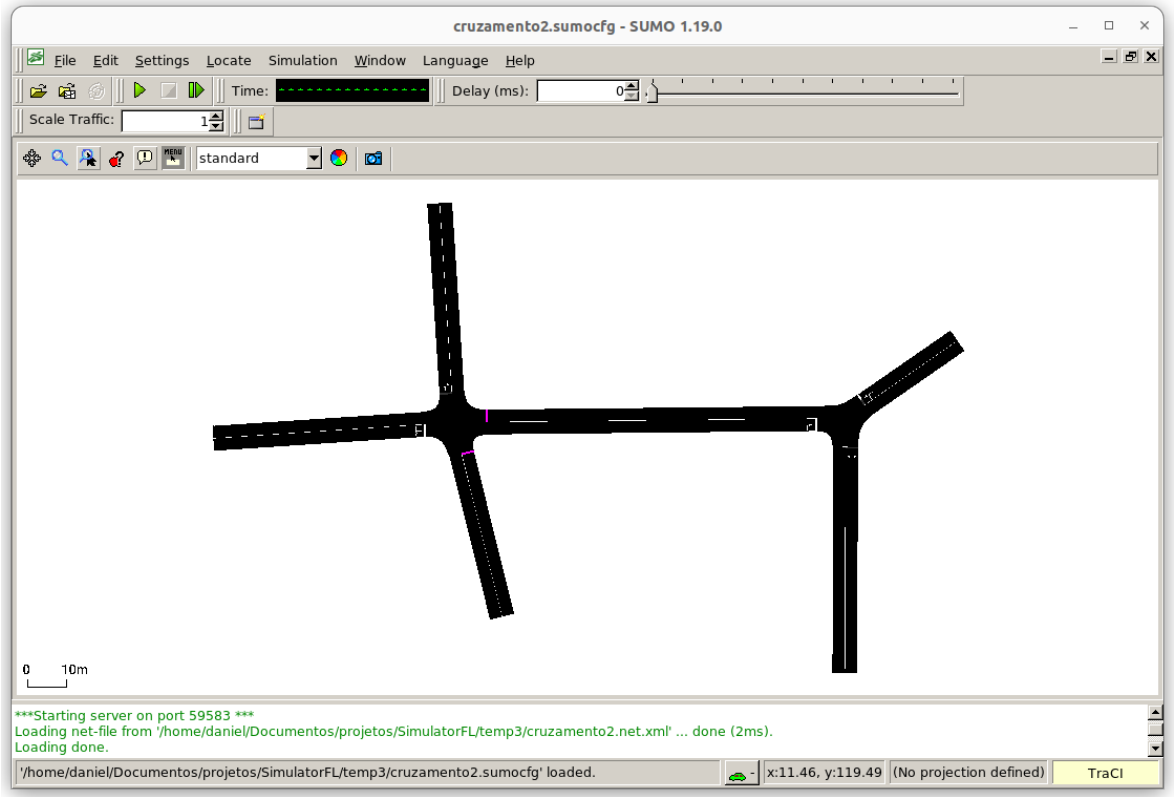


Figura 5.4: Exemplo de mapa do sumo-gui.

zações para correções, especialmente em sua arquitetura. Essas correções não interferem no funcionamento dos módulos já desenvolvidos e descritos na Seção 5.1, mas podem facilitar ainda mais a expansão desses módulos. Nesse sentido, destaca-se a necessidade do desenvolvimento de uma documentação robusta do simulador, assim como de material didático de divulgação da ferramenta (como um site). Essas melhorias serão propostas como trabalhos futuros.

Capítulo 6

Experimentos

Neste capítulo, o algoritmo MoSimFeL é avaliado por meio de experimentos executados no FLSimulator. Todavia, é importante enfatizar que as simulações apresentadas não abrangem todas as funcionalidades implementadas no simulador, especialmente a capacidade de extensão de seus módulos.

Para analisar o MoSimFeL, os algoritmos FedAvg, MoFeL e SimFeL também serão executados em cada cenário proposto permitindo a comparação dos algoritmos a partir dos resultados do treinamento federado. Isso permite uma análise da eficiência desses algoritmos.

Os algoritmos MoFeL e SimFeL são variações simplificadas do MoSimFeL e foram apresentados na Seção 4.1. No MoFeL, são considerados apenas os aspectos de mobilidade, excluindo qualquer avaliação de similaridade prevista no MoSimFeL. Por sua vez, no SimFeL, considera-se apenas os aspectos de similaridade, excluindo qualquer análise de mobilidade prevista no MoSimFeL. A perspectiva de analisar comparativamente esses algoritmos é avaliar como os aspectos de mobilidade e similaridade interferem separadamente na coordenação do treinamento federado.

Ao todo, 4 cenários de simulação são propostos e executado e o restante desse capítulo é organizado da seguinte forma: na Seção 6.1, apresenta-se uma contextualização do cenários simulados, detalhando as variáveis de simulação, configuração do cenário e da distribuição de dados; nas Seções 6.2, 6.3, 6.4 e 6.5, descreve-se os resultados de cada cenário simulado, avaliando-os comparativamente; Por fim, na Seção 6.6, analisam-se conjuntamente os resultados de todos os cenários, discutindo as principais conclusões sobre a avaliação dos experimentos e algoritmos de coordenação do treinamento federado.

6.1 Ambiente Experimental

A fim de contextualizar o ambiente em que os experimentos computacionais foram realizados, destaca-se que eles foram executados em três computadores distintos, com especificações descritos na Tabela 6.1.

	Processador	**Memória Ram (GB)**	**Placa de vídeo**
Computador 1	*Intel i7-7700 3.60 GHz*	*32*	*GeForce GTX 1060 6 GB*
Computador 2	*Intel i5-10500H 2.50 GHz*	*32*	*GeForce GTX 1650*
Computador 3	*Intel i5-12400F 4.5 GHz*	*32*	*Radeon RX6600 XT*

Tabela 6.1: Configuração de computadores usados para executar as simulações.

Apesar da descrição dos computadores na Tabela 6.1, é importante ressaltar que os experimentos não analisam a eficiência computacional dos algoritmos, como a complexidade computacional ou o tempo de execução, limitando-se à análise do comportamento e dos resultados de treinamento. Dessa forma, os resultados apresentados não sofrem influência do ambiente em que o simulador foi executado. Este ambiente só interfere no tempo necessário para a conclusão da simulação. O uso de mais de um computador é justificada para a execução de mais de um cenário simultaneamente, agilizando a conclusão do conjunto de simulações apresentadas nesse trabalho.

Todos os cenários simulados executam uma aplicação de classificação de imagens a partir de uma CNN. A arquitetura CNN para essa aplicação é composta essencialmente pelas as seguintes camadas: *Conv2d(1,10), Conv2d(10,20), Dropout2d, Linear(320, 50), Linear(50, 10).*

A base de dados usada no módulo FLSDistrbuitor são oriundas do MNIST [59]. Essa base de dados é formada por amostras de imagens digitalizadas de dígitos manuscritos por humanos, conforme descrito na seção 2.2. Ao todo há 10 classes, representando os dígitos $[0-9]$.

Ao todo, foi definido que cada cliente selecionado para o treinamento local treina o modelo por 15 épocas, independente do resultado do treinamento local, da sua acurácia ou função de erro. Além disso, cada servidor central define que a taxa mínima da acurácia do

modelo, considerando o conjunto de testes dos clientes de seu domínio, deve permanecer acima de $0,9$. Por tanto, durante a simulação, os servidores centrais solicitam aos clientes dos seus respectivos domínios que avaliem a acurácia do modelo a partir da sua base de dados para teste. Os clientes executam essa avaliação e informam ao servidor central a acurácia que por sua vez calcula a média e o desvio padrão dos resultados e define um veredito sobre a percepção dos clientes. Além disso, o valor de tamanho do lote usado em todos os treinamentos locais será 32.

Ao identificar que essa média está abaixo da acurácia mínima ou que o desvio padrão está acima do limiar máximo, o servidor central deve iniciar o processo de retreinamento do modelo. Não há limites para a quantidade de ciclos que cada servidor central pode executar para alcançar esse requisito da aplicação. Todavia, restringe-se que cada servidor central só pode iniciar até 1 ciclo de treinamento em janela de tempo de 100 passos de simulação.

Ainda sobre os modelos treinados, cada servidor central é contemplado com uma instância aleatória do modelo no início da simulação, ou seja, que foi parametrizada sem um treinamento. Durante os 100 primeiros passos da simulação, os servidores centrais são impedidos de executar o treinamento federado, a fim de que os clientes montem minimamente uma base de dados para que possar ser usada no treinamento local. Assim, nos primeiros instantes após esse intervalo, cada servidor tenta executar o treinamento do seu modelo, pois a definição aleatória do modelo implica na incapacidade de atender aos requisitos de acurácia da aplicação, independentemente da distribuição inicial dos clientes que avaliem esse modelo.

As simulações apresentam um comportamento aleatório devido à natureza semi-determinística do treinamento de aprendizado de máquina e à distribuição aleatória de dados aos clientes no início da simulação. Todavia, considerando o longo período de simulação, eventos estatisticamente improváveis tendem a ser diluídos. Caso esses eventos realmente ocorram, eles serão devidamente destacados na análise do experimento.

Cada cenário é executado por todos os algoritmos de coordenação, gerando quatro execuções de experimentos distintas. Nas próximas seções, parâmetros distintos e específicos de cada cenário serão abordados. Além disso, os resultados dos experimentos em cada cenário são apresentados logo após descrição dos parâmetros com intuito de facilitar a análise crítica da influência dos parâmetros na simulação.

Resumidamente, as principais características em comum dos 4 cenários e que foram descritos nessa seção são:

1. Um cliente só pode se conectar a um único servidor central.
2. Os modelos de cada servidor central são inicializados de forma aleatória.
3. A aplicação dos cenários executa o reconhecimento de imagens digitalizadas de dígitos manuscritos.
4. A aplicação requer que um modelo mantenha uma acurácia mínima de $0,90$, independentemente do desvio padrão.
5. Se o requisito de acurácia mínima da aplicação não for obedecido, o servidor central deve treinar seu modelo até que alcance êxito nos requisitos do item anterior.

Por fim, resume-se o objetivo de cada cenário a seguir:

1. Cenário 1: Avaliação dos algoritmos em um contexto não-IID, com domínios do problema enviesados e com evento de desvio de aprendizagem.
2. Cenário 2: Avaliação dos algoritmos em um contexto não-IID, com domínios do problema enviesados e com evento de desvio de aprendizagem e distribuição uniforme dos dados.
3. Cenário 3: Avaliação dos algoritmos em um contexto IID, com um domínio de problema único e com evento de desvio de aprendizagem.
4. Cenário 4: Avaliação dos algoritmos em um contexto não-IID, com domínios do problema enviesados, com evento de desvio de aprendizagem em uma simulação de tráfego de uma cidade fiel a um cenário real.

6.2 Cenário 1

No Cenário 1, 4 servidores formam uma estrutura de grafo, conforme descrito no Capítulo 4. Cada vértice está conectado a todos os demais vértices inclusive a si mesmo. 300 clientes

são criados no início da simulação e são aleatoriamente e uniformemente distribuídos entre os domínios de todos os servidores centrais.

O tempo necessário para cada cliente executar o treinamento local é definido aleatoriamente e uniformemente entre o intervalo $[10, 100]$. Destaca-se que, nos experimentos desta tese, omitem-se as unidades de tempo e espaço nos parâmetros de simulação. Justifica-se a omissão pela indiferença dessas unidades no comportamento dos experimentos. Todavia, é importante que mantenham a mesma ordem de grandeza. Por exemplo, ao definir que um cliente necessita de $10horas$ para executar um treinamento local, é importante que os demais parâmetros temporais do experimento também sejam definidos em horas. Considerando isso e com o intuito de facilitar a leitura desta tese, em alguns trechos do texto, adotou-se a unidade de tempo como sendo *passos*.

A velocidade de cada cliente é definida aleatoriamente a partir de 3 conjuntos da seguinte maneira: 100 clientes com velocidade entre o intervalo $[1 - 3]$, 100 clientes com velocidade entre $[70 - 100]$ e 100 clientes com velocidade entre $[1 - 100]$. Através dessa distribuição de velocidades, garante-se que haja clientes suficientes para concluir o treinamento local devido sua baixa velocidade ao mesmo tempo que há clientes com alta velocidade. Claramente, os clientes mais lentos tendem a ter maiores chances de concluir o treinamento enquanto os clientes mais rápidos são mais propícios a abandonarem o treinamento por passarem menos tempo em um domínio, considerando que todos se conectaram ao mesmo domínio no mesmo instante.

Em relação aos servidores centrais, para cada servidor é definido uma área. A definição é feita uniformemente e aleatoriamente entre o intervalo $[800, 1200]$. Dessa forma, considerando a velocidade de um cliente c sendo v_c e área de um servidor central s sendo a_s, se c se conectar s, ele deve permanecer em seu domínio por a_s/v_c passos. Após esse período, o cliente pode migrar de domínio com base na topologia definida.

Considerando a estratégia de mobilidade adotada, as rotas de cada cliente são definidas aleatoriamente, incluindo tempo de permanência de conexão em cada servidor central. Dessa maneira, não há erros no cálculo da rota, conhecendo o instante exato de migração de domínios.

Cada cliente recebe amostras para sua base de dados individual ao longo da simulação, armazenando no máximo 2000 amostras. À medida que a base de dados alcança seu tama-

nho máximo, as amostras mais antigas cedem a memória para as amostras mais recentes, mantendo a base de dados atualizada. As amostras são distribuídas aos clientes com base no domínio em que estão conectados. Definiu-se que a cada 50 passos de simulação, os clientes conectados a um servidor central recebem 100 amostras da base de dados considerando o respectivo domínio.

Ao todo, 4 domínios foram definidos. Cada domínio foi enviesado em um subgrupo de classificadores da base de dados, priorizando amostras desse subgrupo. Os subgrupos definidos foram: $[0, 2]$, $[3, 5]$, $[6, 7]$ e $[8, 9]$. Para simular o enviesamento, formou-se uma base se dados para cada domínio. Nessa base de dados, cada domínio manteve 100% das amostras da base de dados do EMNIST dos classificadores pertencentes ao subgrupo priorizado e apenas 50% das amostras referente aos demais classificadores. Após a formação da base de dados, as amostras foram distribuídas aleatoriamente para os clientes.

O domínio inicial de cada servidor é definido sequencialmente e ciclicamente, com base na ordem descrita anteriormente. Assim, o primeiro servidor central assume o primeiro domínio, o segundo servidor assume o segundo domínio e assim por diante. Por sua vez, no quinto servidor central, o ciclo recomeça ao assumir o primeiro domínio. Cada servidor central só tem um domínio.

Ao todo, a simulação dura 2000 passos de simulação e no passo 1000 todos os servidores escolhem aleatoriamente um novo domínio de problema, simulando um desvio de aprendizagem.

Por fim, nesse cenário os dados são distribuídos durante todo o ciclo de vida da simulação com base no servidor central corrente que o cliente está conectado. Ou seja, dado o domínio do servidor central conectado ao cliente, esse domínio atribui aleatoriamente amostras do seu conjunto para cada cliente, como mencionado anteriormente.

6.2.1 Resultados

Iniciando a análise dos resultados da simulação do Cenário 1, na Figura 6.1, apresenta-se a variação da quantidade de clientes conectados em cada servidor central com o passar do tempo. Através dessa figura, é possível concluir que em todos os servidores há migração de clientes, conectado e se descontando de cada servidor. É possível identificar isso através da constante movimentação de todas as curvas do gráfico, que muda de orientação diversas

vezes durante o ciclo de simulação.

Ainda sobre a Figura 6.1, é importante destacar que a mudança de algoritmo de coordenação de treinamento não interfere nesse gráfico, pois a mobilidade dos dispositivos não é influenciada pela aplicação de AF. Ou seja, a a Figura 6.1 representa a migração dos clientes para todos os experimentos do Cenário 1.

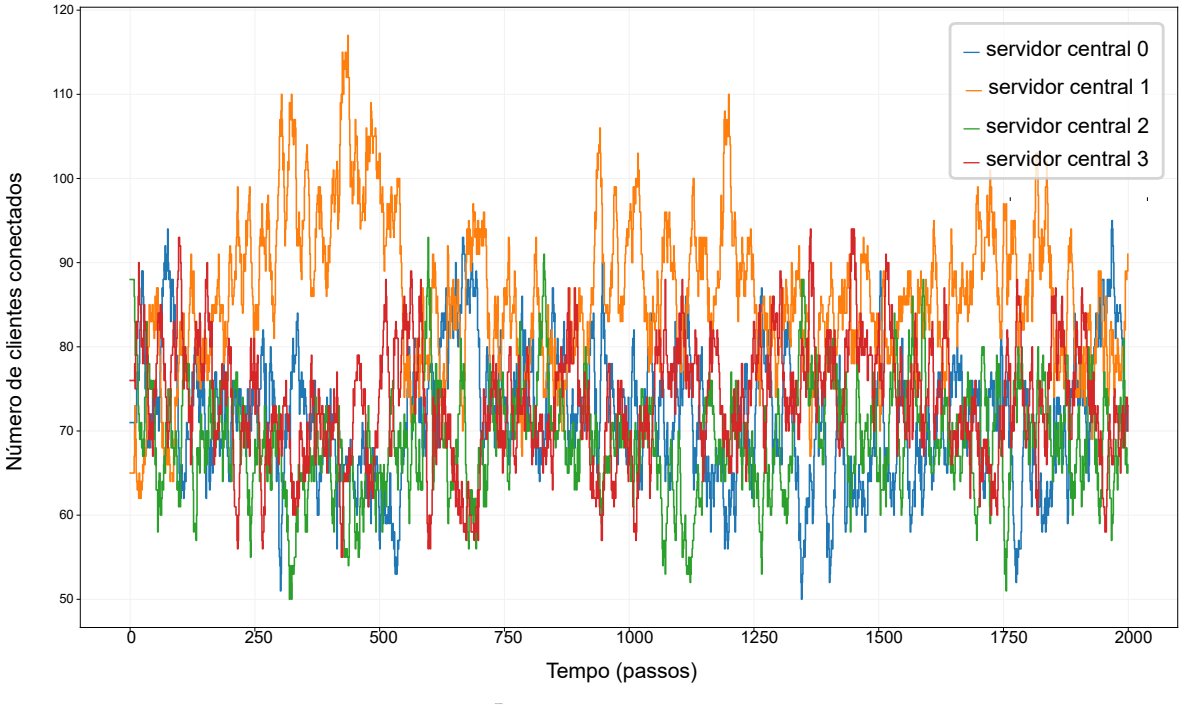
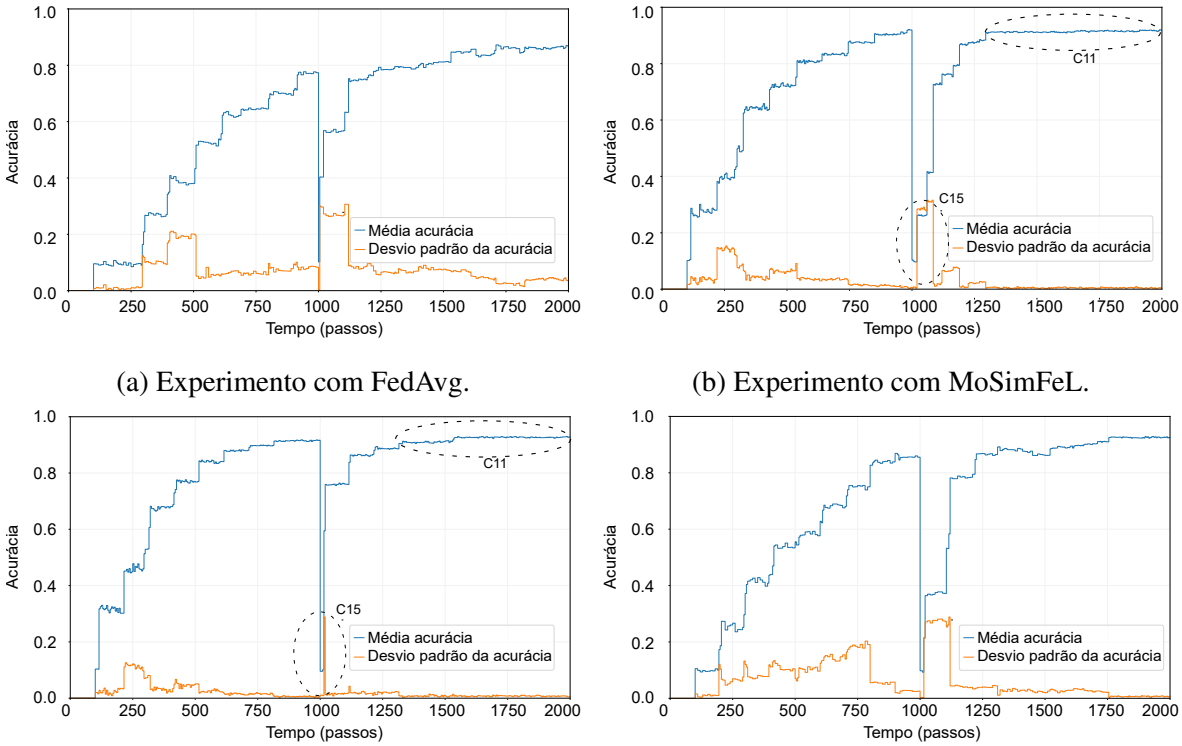


Figura 6.1: Gráfico de conexões de clientes para cada servidor central referente ao Cenário 1.

Nos gráficos da Figura 6.2, é apresentado média e o desvio padrão da avaliação da acurácia dos modelos de todos os servidores centrais a partir da base de dados de teste dos clientes ao longo da simulação. Nessa figura, cada gráfico representa um experimento, ou seja, a simulação do cenário descrito anteriormente a partir de um algoritmo de coordenação de AF.

Através dessa figura, é possível identificar que o modelo não está adequado para a aplicação no início da simulação de nenhum experimento, apresentando uma acurácia baixa. Esse comportamento é esperado mediante a natureza aleatória da criação o modelo no início da simulação, ou seja, sem aprendizado.

Comparando os algoritmos, é possível identificar que no instante anterior ao 1000 apenas o algoritmo FedAvg não conseguiu convergir para uma acurácia média próxima a $0,90$



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.2: Média e desvio padrão da acurácia dos modelos de todos os servidores centrais calculada pelos clientes referente ao Cenário 1.

ficando abaixo disso, com exceção próximo ao final da simulação. Isso aconteceu devido ao número de abandonos de clientes no experimento relativo a ele. Ao todo, foram 1219 abandonos de clientes, conforme apresentado na Tabela 6.2. Os abandonos aconteceram devido a desconsideração da rota dos clientes.

Em relação a curva do desvio padrão da acurácia (Figura 6.2), nota-se que o MoSimFeL e o SimFeL apresentaram um comportamento similar, enquanto o SimFeL e o FedAvg apresentaram maior taxa de desvio. O marcador $C15$ destaca a região em que o MoSimFeL apresentou um maior desvio padrão que o MoFeL, após o desvio de aprendizagem no instante 1000. Essa diferença é justificada pela recuperação do desvio de aprendizagem de alguns servidores centrais ser mais rápida do que a de outros, no cenário do MoSimFeL. É importante ressaltar que o algoritmo MoSimFeL alcançou a acurácia mínima de $0,9$ após o desvio de aprendizagem, antes do MoFeL.

Em relação as curvas do desvio padrão da acurácia do SimFeL e FedAvg (Figura 6.2), os valores mais altos (comparados ao MoFeL e MoSimFeL) indicam que esses algoritmos

Algoritmos	QA	QT
FedAvg	1355	37
MoSimFeL	0	50
MoFeL	0	52
SimFeL	985	55

Tabela 6.2: Número de abandonos e ciclos concluídos referente ao Cenário 1.

obtiveram resultados melhores em alguns servidores centrais em detrimento de outros, apresentando heterogeneidade na avaliação da eficiência do modelo.

Não houve abandono de clientes nas simulações com os algoritmos MoSimFeL e MoFeL, pois esses algoritmos escolheram clientes que não migraram de domínio antes de finalizar o treinamento do modelo. Para isso, os algoritmos consideraram a rota dos clientes e calcularam o tempo restante de cada cliente no domínio de um servidor central. Dessa maneira, todos os treinamentos inicializados por um servidor central foram concluídos.

Em contrapartida, o FedAvg não considera a rota dos clientes e, por isso, selecionou clientes incapazes de concluir o treinamento, acarretando em 1355 abandonos de treinamento. O SimFeL também não considera a rota e, por isso, também apresentou abandono de treinamento por clientes que migraram de rede durante um ciclo de treinamento federado. Ao todo, 985 clientes abandonaram o treinamento. Todavia, considerando as curvas apresentadas na Figura 6.2, conclui-se que os ciclos de treinamento do SimFeL foram significativamente melhores do que os do FedAvg mesmo com os abandonos. Isso ocorre devido a qualidade de cada ciclo de treinamento executado. Como no algoritmo SimFeL a base de dados de cada cliente é indiretamente avaliada, os clientes que não abandonaram o treinamento conseguiram contribuir mais significativamente para o modelo global.

Ainda sobre o número de abandonos de treinamento no algoritmo SimFeL, identifica-se que esse algoritmo teve menos abandonos do que o FedAvg (370 ciclos a menos, conforme Tabela 6.2). Isso ocorre, porque o SimFeL indiretamente identifica clientes com base de dados que ainda não foram abordados no treinamento. Esses clientes possivelmente são oriundos de outros domínios e se conectaram ao servidor central há menos tempo do que clientes com bases de dados já contempladas. Assim, esses clientes recentes tendem a ficar

mais tempo conectados ao servidor central do que os demais clientes, aumentando as chances de concluir o treinamento e diminuindo a taxa de abandono. Dessa maneira, é possível afirmar que, o SimFeL conseguiu avaliar aspectos de mobilidade de forma indireta, diminuindo a taxa de abandono de clientes. Nesse sentido, o número elevado de abandonos do treinamento no FedAvg inviabilizou diversos ciclos de treinamento, com 18 ciclos a menos que o algoritmo com maior número de treinamento, o SimFeL. Todavia, a avaliação indireta do SimFeL conseguiu viabilizar o treinamento de diversos ciclos, mesmo com uma menor qualidade que o MoFeL e MoSimFeL.

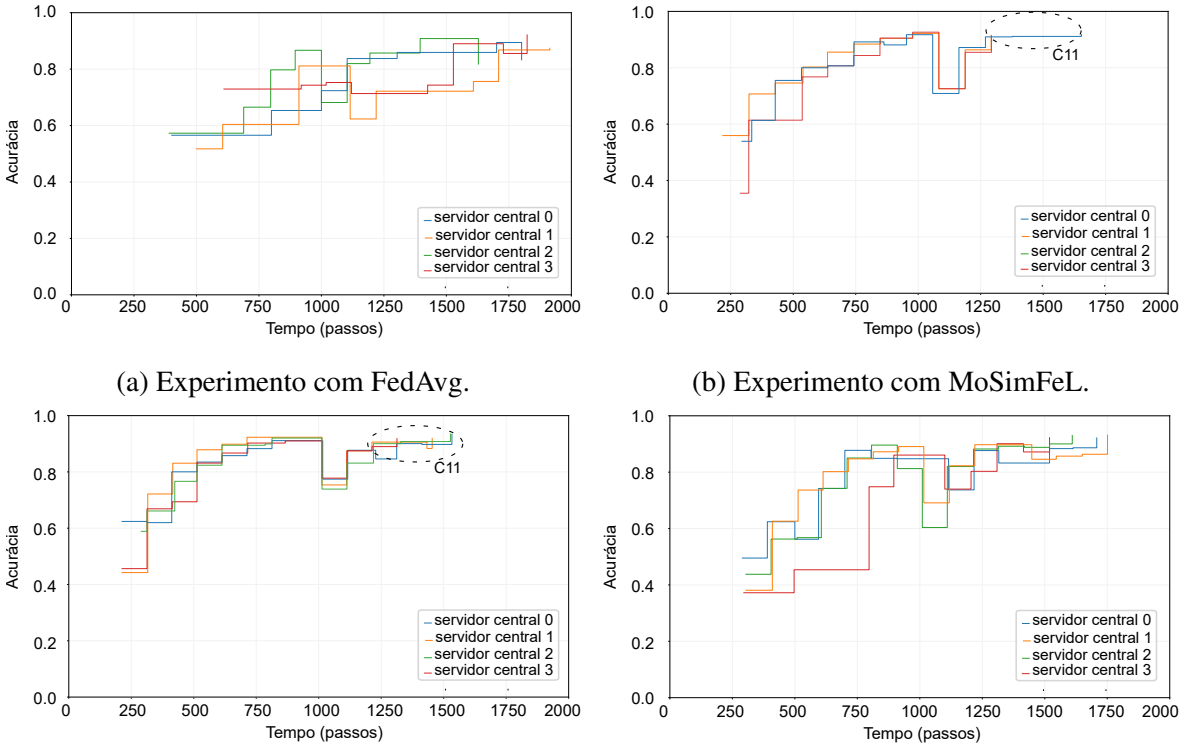
Na Figura 6.3, é apresentado o gráfico do valor de ACC_A pelo tempo. Nesse gráfico, cada curva é análise de ACC_A de cada servidor central. Através dos gráficos, observa-se que os algoritmos de FedAvg e SimFeL apresentaram resultados mais baixo, devido a ineficiência de cada ciclo de treinamento, devido a ocorrência de abandonos que influenciava diretamente no tamanha da base de dados usada no treinamento. Por sua vez, nos algoritmos MoSimFeL e MoFeL, cada ciclo de treinamento foi otimizado pois todos os clientes selecionados contribuíram para o treinamento, repercutindo em valores de ACC_A maiores do que os algoritmos SimFeL e FedAvg.

A análise conjunta dos gráficos das Figuras 6.3 e 6.2 norteia a justificativa para o SimFeL apresentar mais treinamentos que o MoSimFeL e MoFeL. Através do destaque $C11$ nos gráficos 6.3b, 6.3c, 6.2c e 6.2b, verifica-se que a acurácia do modelo alcançou o mínimo requerido pela aplicação por diversas vezes e isso extinguiu a necessidade de mais treinamentos. Todavia, o algoritmo SimFeL só alcançou esse mínimo próximo ao fim da aplicação requerendo mais treinamentos.

Nas Figuras 6.4 e 6.5, apresenta-se respectivamente o gráfico do valor de ACC_N e ACC_O pelo tempo, para cada um dos experimentos. Em todos os gráficos, cada curva representa a avaliação de um servidor central.

Em relação a esses gráficos, é possível confirmar que o FedAvg obteve poucos treinamentos e, por isso, os gráficos demonstrados apresentam uma quantidade de pontos menor do que os demais. Como os treinamentos nesse algoritmo foram ineficientes, a análise de ACC_A, ACC_O e ACC_N tornam-se inconclusivas visto que a definição dos parâmetros de modelos não convergiu para um aprendizado minimamente aceitável.

Comparando o MoFeL e o MoSimFeL, identifica-se que o algoritmo MoFeL apresentou



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

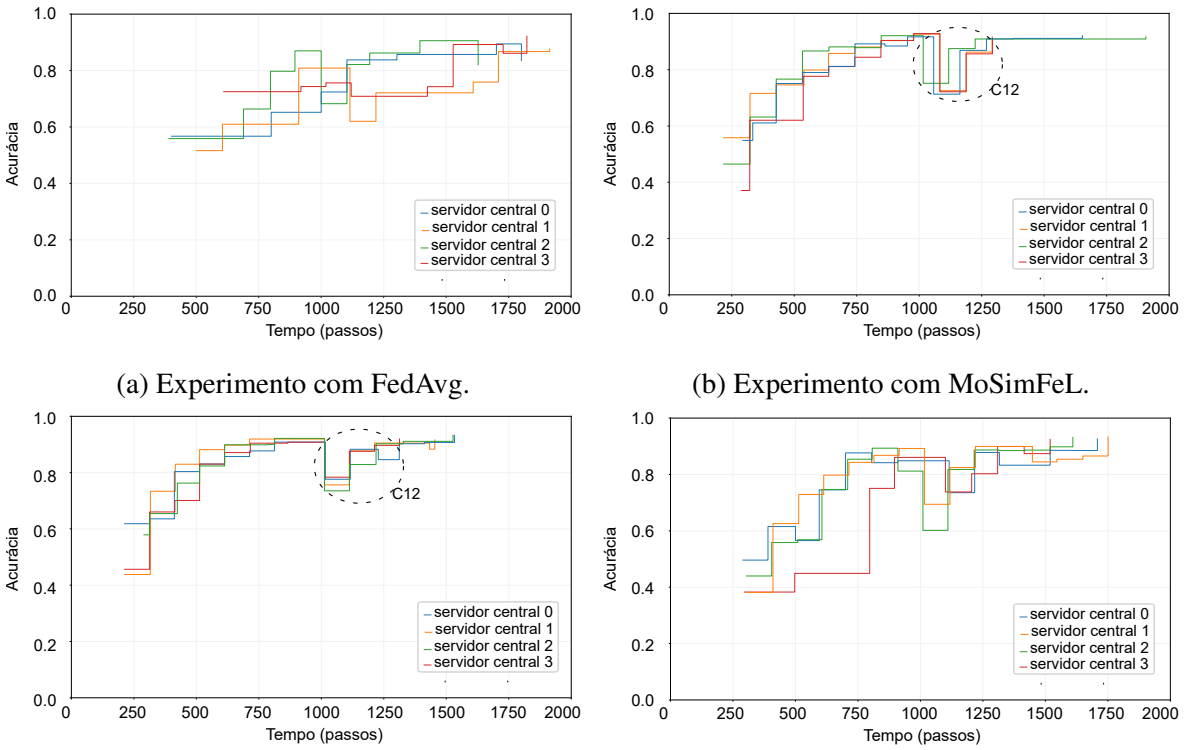
(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.3: ACC_A ao longo da simulação de todos os servidores centrais referente ao Cenário 1.

resultados um pouco melhores para clientes recém conectados ao do domínio de um servidor do que o MoSimFeL. Essa conclusão é inferida através da observação da Figura 6.4, em que as curvas do gráfico 6.4c estão mais próximas do limiar desejado de 0.9 do que as curvas do gráfico 6.4b. Ou seja, o MoFeL conseguiu atender de forma mais satisfatório os clientes recém-chegados ao domínio de cada servidor. Todavia, analisando o treinamento após o desvio de aprendizagem no instante 1000 (destacado pelo marcador $C12$) identifica-se que o MoSimFeL conseguiu recuperar a eficiência do modelo de forma mais equilibrada entre todos os servidores centrais. Ou seja, a similaridade mostrou-se novamente eficiente na análise indireta de bases de dados dos clientes, aumentando a eficiência do treinamento do modelo.

Executando a mesma análise visual sobre os gráficos de ACC_O, é possível observar o comportamento inverso. Ou seja, o MoSimFeL apresentou resultado um pouco melhores do que o MoFeL conseguindo ser mais eficiente na avaliação dos clientes mais antigos ao domínio. Isso ocorre exatamente devido à análise indireta da base de dados dos clientes,



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

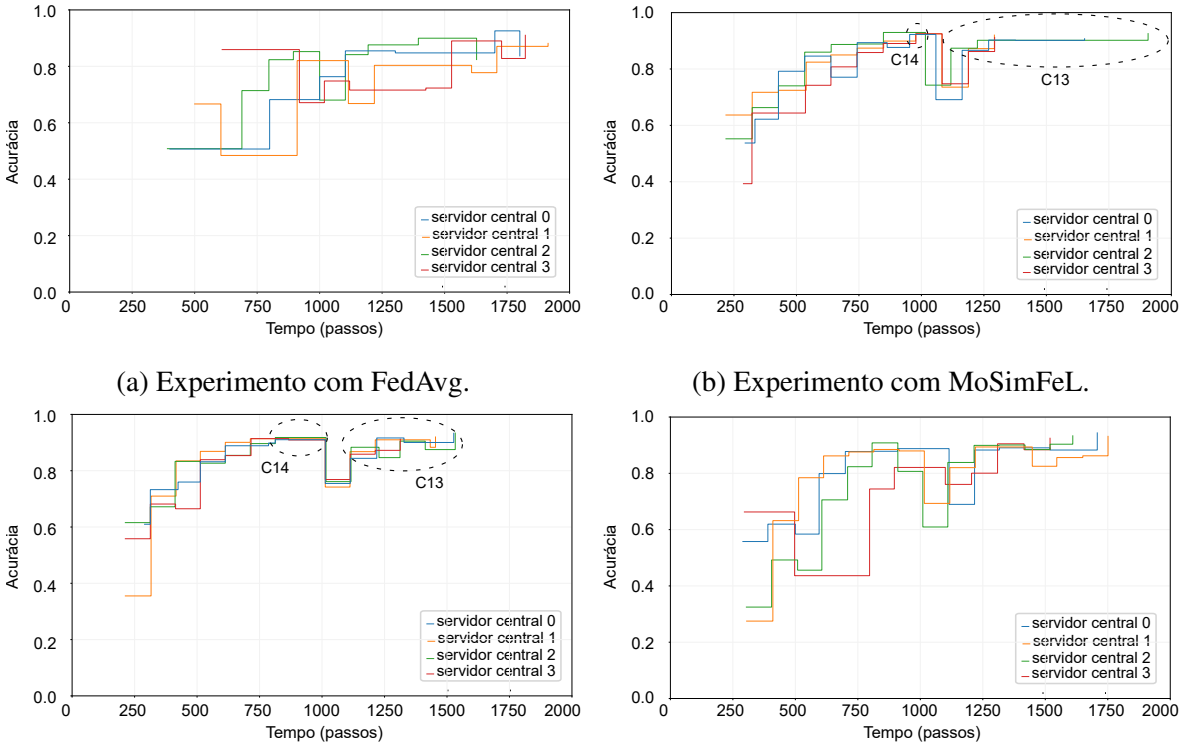
(d) Experimento com SimFeL.

Figura 6.4: ACC_N ao longo da simulação de todos os servidores centrais referente ao Cenário 1.

amenizando o problema de esquecimento do modelo com o retreinamento a partir de novas bases de dados inteiramente distintas das anteriores.

Destaca-se dois marcadores nos gráficos 6.5c e 6.4b, respectivamente do MoFeL e MoSimFeL. O primeiro marcador $C13$ destaca que o MoSimFeL conseguiu uma melhor convergência de acurácia (considerando os clientes intrínsecos a métrica ACC_O) em seus últimos treinamentos, em relação ao MoFeL, devido a menor dispersão das curvas. O segundo marcador $C14$ destaca os últimos treinamentos de cada servidor central antes do desvio de aprendizagem. Nesses treinamentos, o MoSimFeL conseguiu uma acurácia maior do que o MoFeL considerando os clientes previstos na métrica ACC_O.

Em relação ao SimFel, os gráficos 6.4d e 6.5d indicam uma grande dispersão entre as curvas de cada servidor central. Isso ocorre pela baixa participação de clientes no treinamento em decorrência do abandono de treinamento.



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.5: ACC_O ao longo da simulação de todos os servidores centrais referente ao Cenário 1.

6.3 Cenário 2

O Cenário 2 é similar ao Cenário 1, com exceção da técnica de distribuição de dados. No Cenário 2, as amostras são distribuídas para um cliente apenas no primeiro passo da simulação, sendo suficientes para preencher a memória reservada para sua base de dados. Portanto, a base de dados de um cliente permanece inalterada e com o enviesamento definido unicamente pelo primeiro servidor central ao qual esse cliente se conecta, mesmo que altere de domínio à medida que migre de conexão com um servidor central.

6.3.1 Resultados

Assim como no cenário anterior, a análise deste cenário é iniciada a partir da observação do gráfico do número clientes conectado a cada servidor de borda ao longo do tempo (Figura 6.6). Identifica-se visualmente que durante toda a simulação ocorreram migrações, como no Cenário 1.

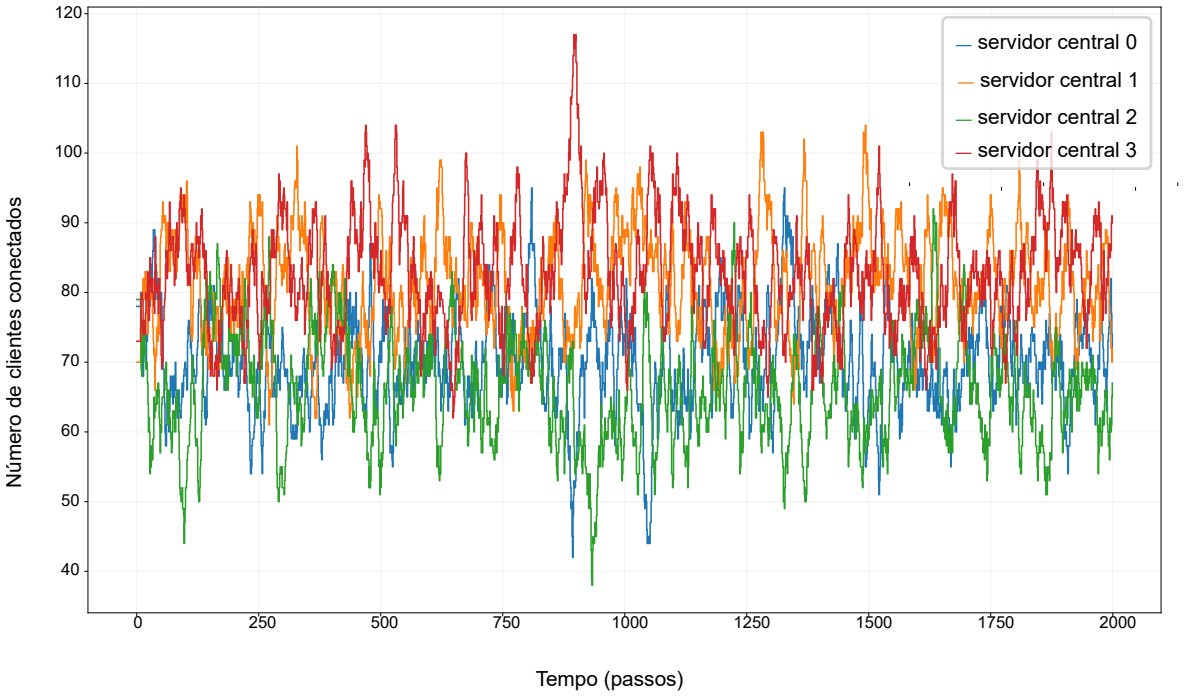
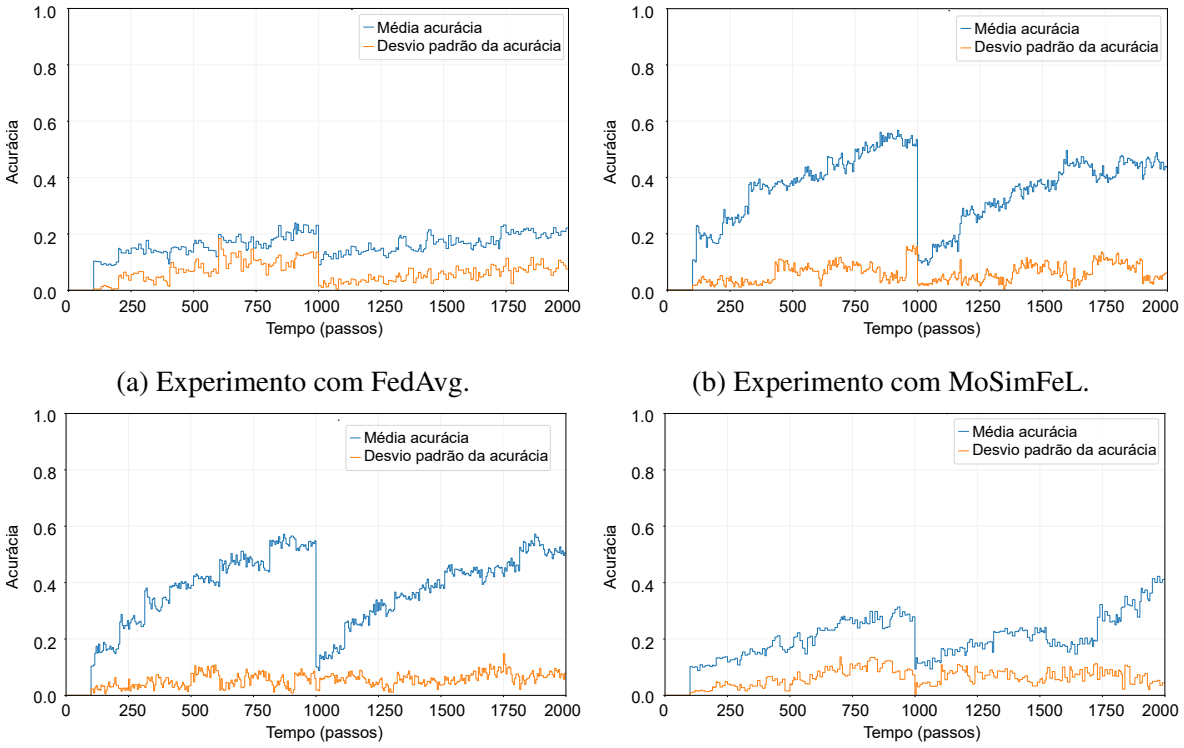


Figura 6.6: Gráfico de conexões de clientes para cada servidor central referente ao Cenário 2.

No gráfico da Figura 6.7, apresenta-se a média e o desvio padrão da avaliação da acurácia dos modelos ao longo do experimento. Através desses gráficos é possível identificar uma pioria nos resultados de todos os algoritmos, comparado ao cenário anterior.

Para justificar a piora entre os Cenários 1 e 2, é preciso compreender que o domínio de cada servidor central é estático em dois intervalos ($[0, 1000[$ e $[1000, 2000]$) e que a base de dados de cada cliente também é estática. No início da simulação, os clientes recebem uma base de dados que representa o domínio do servidor central ao qual estão conectados. Como essa base de dados inicial é desbalanceada, há um favorecimento de amostras para o primeiro domínio do cliente. Esse conjunto de amostras inicial representa de forma precisa o domínio de cada servidor.

Todavia, com a migração de domínios em decorrência da movimentação dos clientes, essa base de dados inicial também migra. Dessa maneira, após um curto período, cada servidor central está conectado a um arranjo de clientes com enviesamento de outros domínios, desfavorecendo o treinamento de um modelo especializado no domínio de seu problema. Com isso, não há ciclos de treinamento suficientes para que o modelo mantido por um ser-



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.7: Média e desvio padrão da acurácia dos modelos de todos os servidores centrais calculada pelos clientes referente ao Cenário 2.

vidor central aprenda o domínio desse servidor, repercutindo negativamente na eficiência da aplicação durante a simulação.

Na Tabela 6.3, apresenta-se o número de abandonos de treinamento para cada um dos cenários. É possível observar que os algoritmos que não consideram a mobilidade (SimFeL e FedAvg) obtiveram taxas de abandono maior que zero.

É importante enfatizar que os clientes que abandonaram o treinamento empenharam recursos desnecessariamente. Em uma aplicação real e com um ciclo de vida prolongado, a execução de treinamentos locais é um desafio. Assim, o aumento de abandonos pode comprometer a disposição dos clientes em contribuir com o treinamento e, consequentemente, resultar no esvaziamento de clientes participantes de um ciclo de treinamento.

Analisando conjuntamente a Figura 6.7 e a Tabela 6.3, especificamente os algoritmos SimFeL e FedAvg, é possível identificar que a análise de similaridade teve efeitos positivos, mesmo sem a avaliação de mobilidade. O número de treinamentos concluídos pelo SimFeL foi o dobro do FedAvg, além de 109 abandonos de treinamento a menos.

Algoritmos	QA	QT
FedAvg	1421	27
MoSimFeL	0	72
MoFeL	0	77
SimFeL	1312	54

Tabela 6.3: Número de abandonos e ciclos concluídos referente ao Cenário 2.

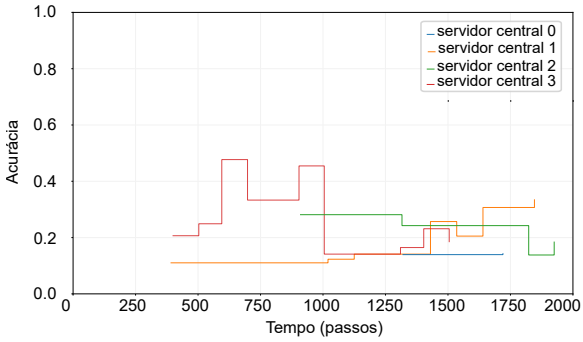
Na Figura 6.8, apresenta-se o gráfico do valor de ACC_A ao longo do tempo. Nesse gráfico, cada curva representa a análise de ACC_A de cada servidor central. Através dos gráficos, é possível observar que os algoritmos FedAvg e SimFeL apresentaram resultados mais baixos, devido à ineficiência de cada ciclo de treinamento, causada pela ocorrência de abandonos que influenciavam diretamente no tamanho da base de dados usada no treinamento. Por sua vez, nos algoritmos MoSimFeL e MoFeL, cada ciclo de treinamento foi otimizado, pois uma parte maior dos clientes selecionados contribuiu para o treinamento, resultando em valores de ACC_A maiores do que nos algoritmos SimFeL e FedAvg.

Nas Figuras 6.9 e 6.10, apresenta-se respectivamente o gráfico do valor de ACC_N e ACC_O pelo tempo, para cada um dos experimentos. Em todos os gráficos, cada curva representa a avaliação de um servidor central.

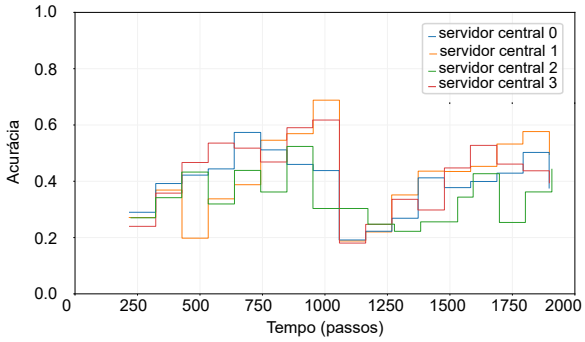
Analisando das Figuras 6.9 e 6.10 é possível observar que as curvas dos servidores centrais nos experimentos com os algoritmos MoFeL e MoSimFeL são mais próximas uma das outras quando comparado aos experimentos com os algoritmos FedAvg e SimFeL. Novamente, isso é justificado pelo volume de dados usados no treinamento e a quantidade de ciclos de treinamento executados.

Por mais que o requisito de acurácia mínima da aplicação não tenha sido alcançado em nenhum dos experimentos, os algoritmos MoFeL e MoSimFeL mostram-se eficientes ao manter o aprendizado de lições antigas à medida que assimilam novos aprendizados. Isso é comprovado comparando visualmente o comportamento das curvas das Figuras 6.9 e 6.10. É possível observar que as métricas ACC_N e ACC_O de cada um dos algoritmos apresentaram um comportamento de tendência e são mais agrupadas entre si do que os demais algoritmos.

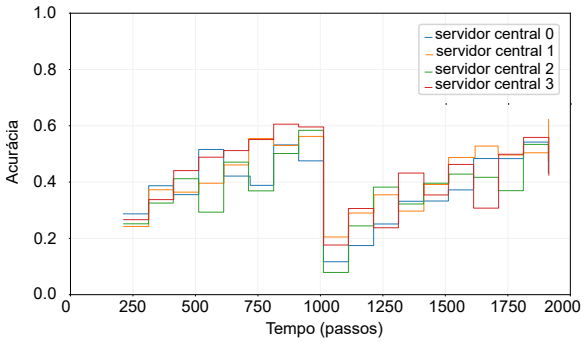
Especialmente as curvas referentes ao FedAvg, em todas as métricas, representadas nas



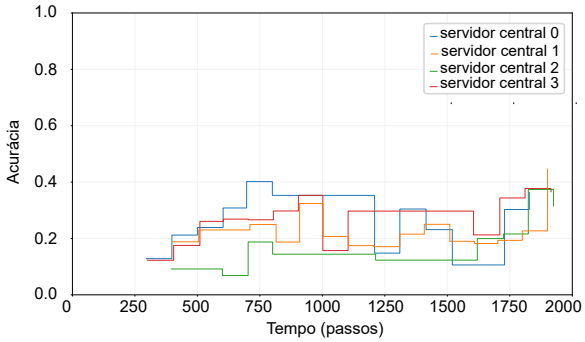
(a) Experimento com FedAvg.



(b) Experimento com MoSimFeL.



(c) Experimento com MoFeL.



(d) Experimento com SimFeL.

Figura 6.8: ACC_A ao longo da simulação de todos os servidores centrais referente ao Cenário 2.

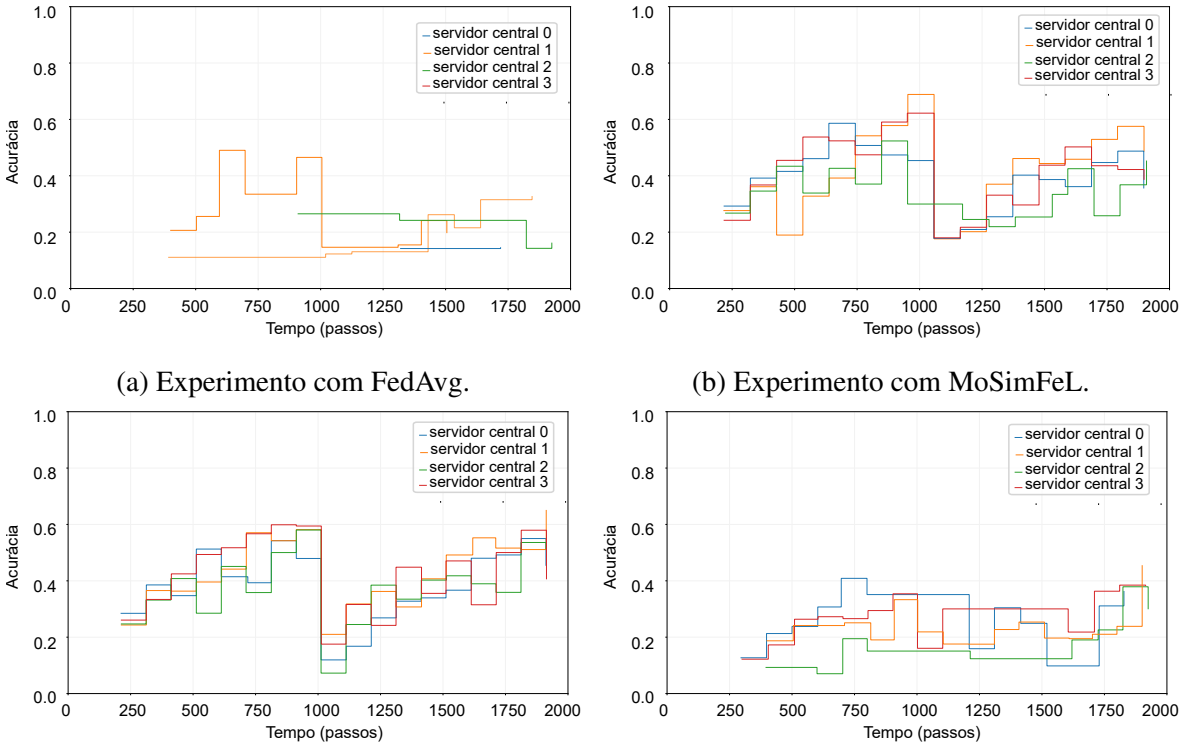
Figuras 6.8a, 6.9a e 6.10a, apresentaram valores dispersos entre os servidores de borda, incluindo ausência de amostras por servidor

6.4 Cenário 3

O Cenário 3 é similar ao Cenário 1, com exceção da definição de multidomínios. No Cenário 3, todos os servidores centrais são especializados no mesmo domínio de problema. Nesse caso, não há enviesamento na distribuição de dados e todos os domínios são especializados no reconhecimento dos dígitos de $[0-9]$.

6.4.1 Resultados

Assim como nos cenários anteriores, a análise desse cenário é iniciada a partir da observação do gráfico do número clientes conectado a cada servidor de central ao longo do tempo



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

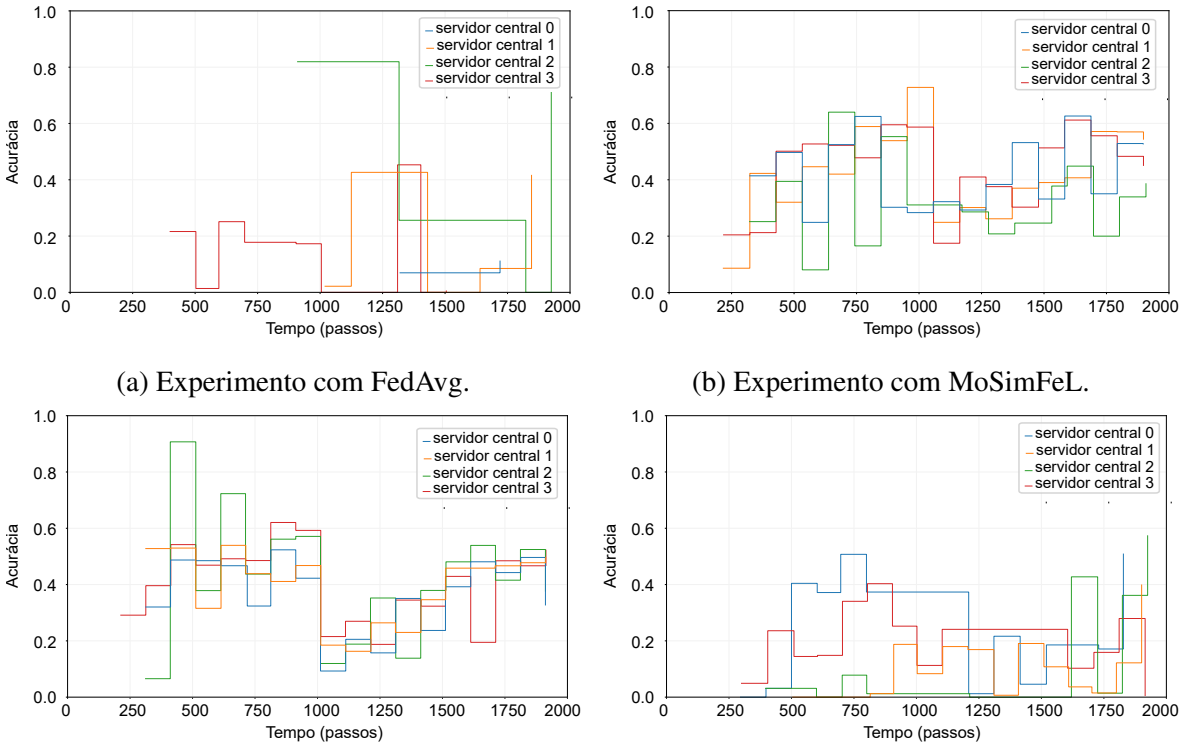
Figura 6.9: ACC_N ao longo da simulação de todos os servidores centrais referente ao Cenário 2.

(Figura 6.11). A análise da Figura 6.11 atesta que houve migração de clientes durante todo o experimento, como no Cenários 1 e 2.

No gráfico da Figura 6.12 é apresentado a média e o desvio padrão da avaliação da acurácia dos modelos ao longo do experimento.

Os gráficos da Figura 6.12 comprovam que os algoritmos MoSimFeL, SimFeL e MoFeL alcançaram a acurácia mínima requerida pela aplicação. Todavia, o MoFeL e o MoSimFeL alcançaram esse feito de forma mais rápida. Especificamente após o desvio de aprendizagem, no SimFeL, a média de avaliação da acurácia só indicou a acurácia mínima no final da simulação. Mesmo assim, como o desvio padrão dessa região é maior que zero, há indícios de que nem todos os servidores centrais alcançaram essa taxa. Isso comprova a importância de avaliar a mobilidade dos clientes, mesmo em cenários sem múltiplos domínios.

Os algoritmos FedAvg e SimFeL apresentaram curvas de desvio padrão com valores mais altos do que o MoFeL e SimFeL, indicando uma avaliação heterogênea dos modelos de cada servidor central, como presenciado no Cenário 1. Especialmente o FedAvg apresentou, du-



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.10: ACC_O ao longo da simulação de todos os servidores centrais referente ao Cenário 2.

rante a simulação, as taxas de maior desvio e, considerando a média da acurácia, conclui-se a incapacidade desse algoritmo em manter um treinamento eficiente para todos os servidores centrais em cenários sem múltiplos domínios, mas com mobilidade.

Na Tabela 6.4 exibe-se o número de ciclos de treinamento executados em cada cenário. Observa-se um comportamento similar as Cenários 1 e 2 em que os algoritmos MoSimFeL e MoFeL obtiveram um maior número ciclos de treinamento comparado ao FedAvg. É compreensível esse comportamento similar aos cenários anteriores devido ao fator limitante do treinamento ser a mobilidade e, no cenários 1, 2, 3, as características de mobilidade foram a mesmas. Ainda comparando os cenários, o SimFeL e o FedAvg conseguiu realizar mais treinamentos no Cenários 3 do que em cenários anteriores. Isso aconteceu, devido ao arranjo aleatório da simulação que propiciou um maior tempo de conexão de clientes a um servidor de borda em relação aos outros cenários.

Comparando o SimFeL e o FedAvg, o algoritmo SimFeL conseguiu um maior número de ciclos de treinamento também nesse cenário, com aproximadamente 11 treinamentos a mais.

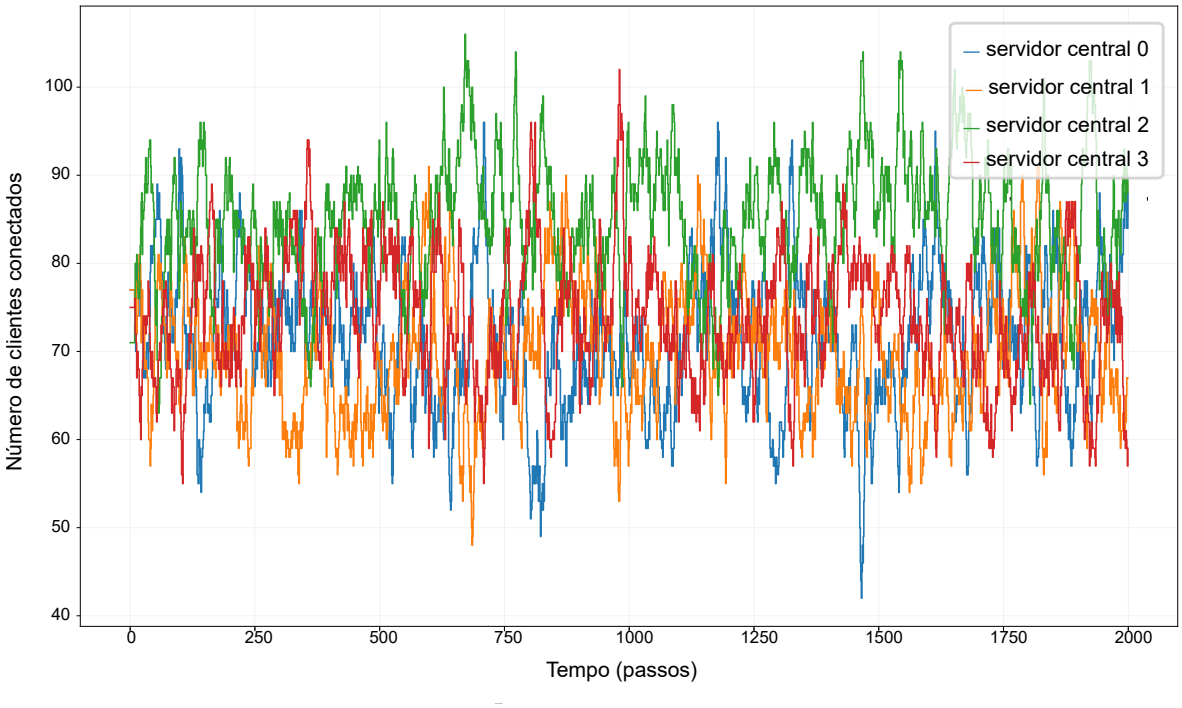


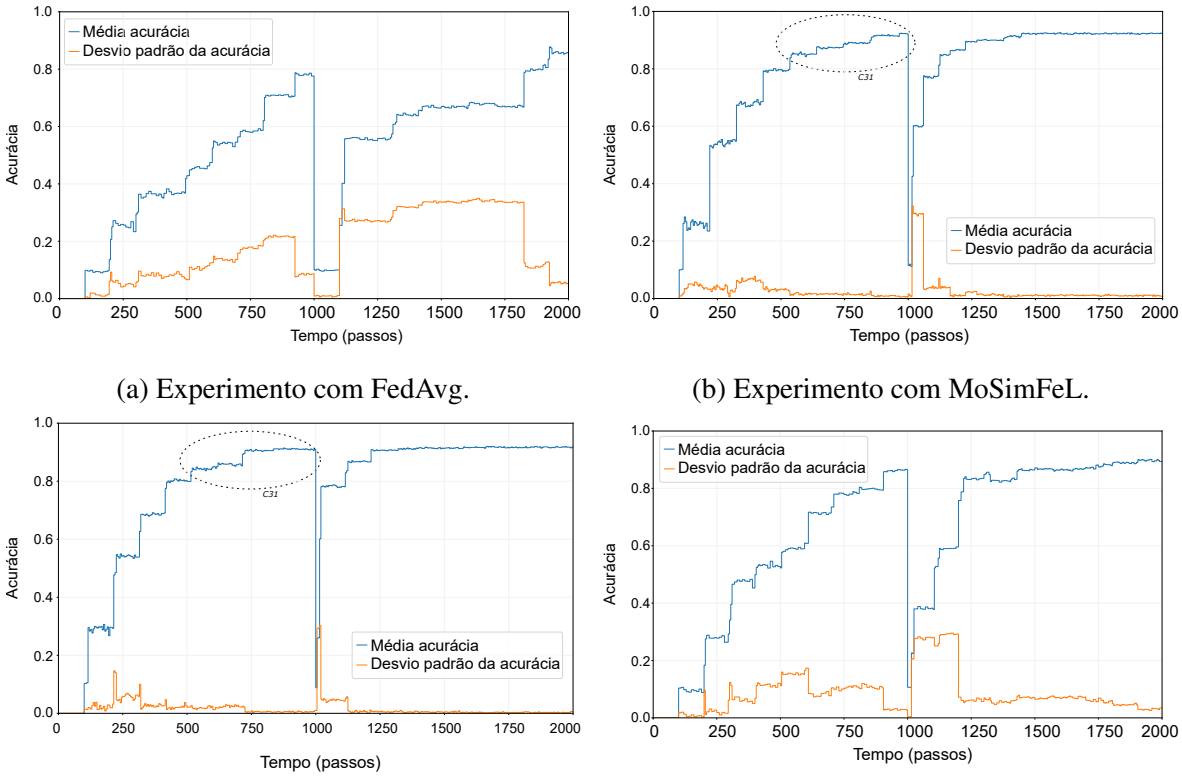
Figura 6.11: Gráfico de conexões de clientes para cada servidor central referente ao Cenário 3.

A justificativa para isso é a mesma do Cenário 2, ou seja, através da análise de similaridade, o SimFeL selecionou mais clientes recém conectados ao domínio do servidor central. Por serem mais recentes, esses clientes devem permanecer por mais tempo no domínio do servidor central, favorecendo a escolha de clientes aptos para concluir o treinamento antes de migrarem.

Essa explicação também respalda a aproximação do número de ciclos de treinamento do SimFeL com o MoFeL e o MoSiFel. A diferença do número de ciclos de treinamento executados entre os algoritmos SimFel e MoFeL foi de 3 ciclos, enquanto a diferença entre

Algoritmos	QA	QT
FedAvg	1378	38
MoSimFeL	0	50
MoFeL	0	46
SimFeL	1046	49

Tabela 6.4: Número de abandonos e ciclos concluídos referente ao Cenário 3.



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

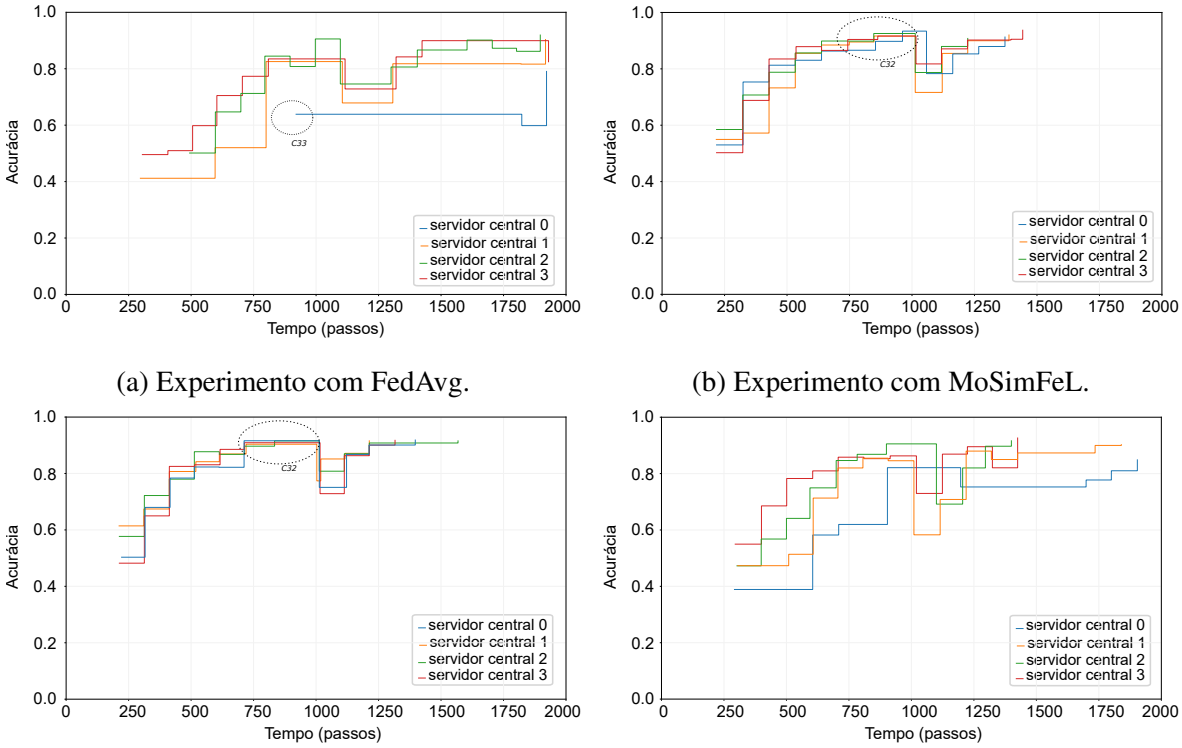
(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.12: Média e desvio padrão da acurácia dos modelos de todos os servidores centrais calculada pelos clientes referente ao Cenário 3.

SimFeL e MoSimFeL foi de 1 ciclo. Todavia, é possível observar que mesmo com o número de treinamentos alto de ciclos de treinamento, o SimFeL não conseguiu aumentar a acurácia de forma significativa como os algoritmos MoFeL e SimFeL, conforme comprovado na Figura 6.12. Na Figura 6.12, observa-se facilmente no intervalo $[500, 1000]$ (destacado nos gráficos, através do destaque $C31$) que os algoritmos MoSimFeL e MoFeL convergiram para a acurácia mínima da aplicação antes do SimFeL. Isso comprova a importância de analisar a disponibilidade dos dispositivos para concluir o treinamento através da mobilidade. Ou seja, avaliar indiretamente a base de dados dos clientes através da similaridade contribui para a eficiência do treinamento. No entanto, não é vantajoso selecionar clientes com boas bases de dados se estes não estiverem disponíveis para concluir o treinamento.

Na Figura 6.13, apresenta-se o gráfico do valor de ACC_A pelo tempo. Nesse gráfico, cada curva é análise de ACC_A de um servidor central. Através da Figura 6.13, observa-se que os algoritmos MoSimFeL e MoFeL obtiveram curvas de treinamento mais próximas entre



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

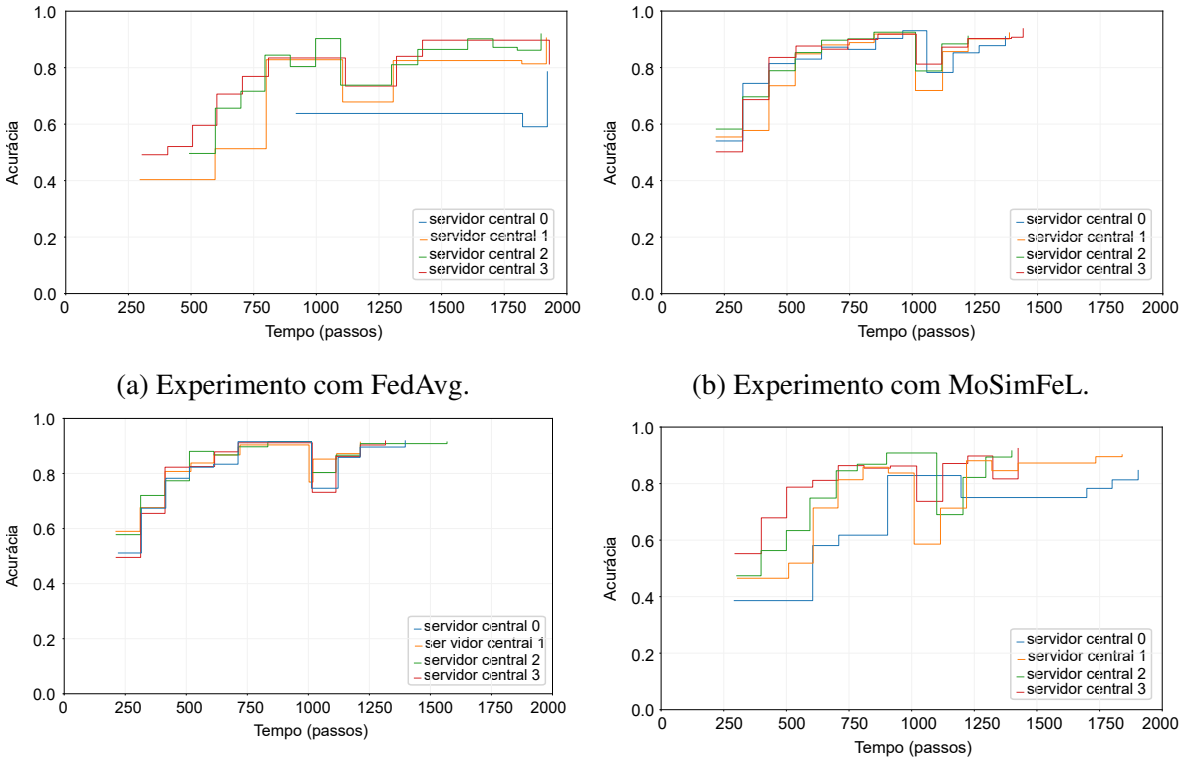
(d) Experimento com SimFeL.

Figura 6.13: ACC_A ao longo da simulação de todos os servidores centrais referente ao Cenário 3.

si, ou seja, menos dispersas, conforme exemplificado em um intervalo no destaque $C32$. Novamente, isso demonstra que os treinamentos executados através desses algoritmos foram mais efetivos. Além disso, destaca-se a incapacidade do FedAvg de executar treinamentos em cenários com mobilidade através da análise do *servidor central 0*. Esse servidor só conseguiu executar seu primeiro treinamento no intervalo $[750, 1000]$, destacado por $C33$.

As Figuras 6.15 e 6.14, é apresentado o gráfico do valor de ACC_A pelo tempo. Nesse gráfico, cada curva representa a métrica ACC_A de cada servidor central.

Em relação a $ACCO$ e $ACCN$ destaca-se que em todos os algoritmos, as curvas dessas duas métricas foram parecidas. Isso é justificado pela homogeneidade da base de dados dos clientes que abordam o mesmo domínio do problema, sem enviesamento. Logo, não há distinção entre clientes recém conectados ou clientes mais antigos de um servidor central. Dessa forma, é possível concluir que os resultados positivos do SimFeL em relação ao FedAvg são em decorrência de sua capacidade de inferir, mesmo que de forma imprecisa, a capacidade de um cliente concluir o treinamento mediante sua disponibilidade.



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

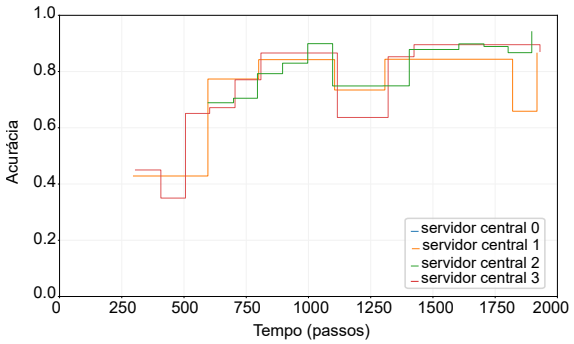
(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

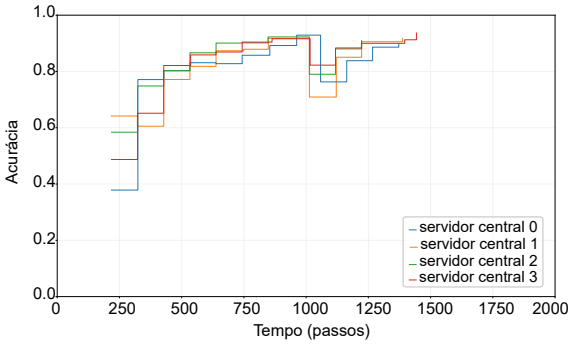
Figura 6.14: ACC_N ao longo da simulação de todos os servidores centrais referente ao Cenário 3.

Nesse caso, o abandono dos clientes não influencia na perda de diversidade da base de dados, pois inicialmente todos os clientes possuem base de dados semelhante. Nesse cenário, o abandono repercute na ausência de amostras dos classificadores que representam o domínio do servidor central.

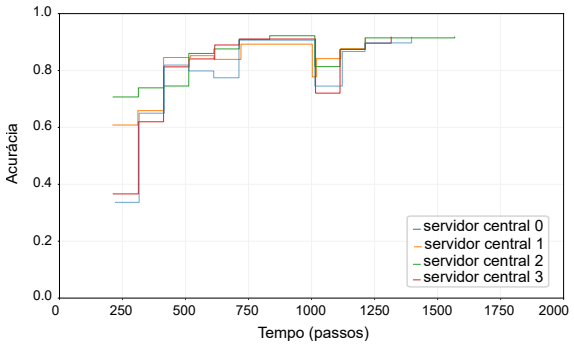
Assim, é possível concluir que a avaliação de mobilidade se demonstrou importante mesmo em cenários com distribuição IID e estacionários, pois apresentou resultados melhores que os demais algoritmos. Contudo, a avaliação de mobilidade demandou empenho de recursos computacionais para avaliação das rotas dos clientes, tanto de processamento e memória ao fazer os cálculos de predição, quanto de banda de comunicação para transmitir essas informações. Assim, é preciso que a aplicação avalie se as melhorias apresentadas pelo MoFeL são justificáveis mediante o aumento do custo computacionais do algoritmo.



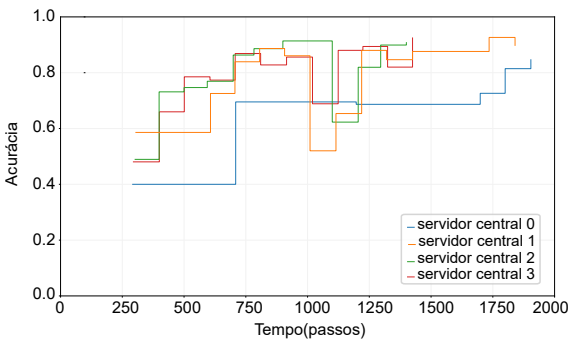
(a) Experimento com FedAvg.



(b) Experimento com MoSimFeL.



(c) Experimento com MoFeL.



(d) Experimento com SimFeL.

Figura 6.15: ACC_O ao longo da simulação de todos os servidores centrais referente ao Cenário 3.

6.5 Cenário 4

No Cenário 4, o SUMO controla a mobilidade e a distribuição de dados é enviesada da mesma forma que no Cenário 1. Define-se de forma arbitrária que a conexão de um cliente com um servidor central é decidida a partir da aproximação geográfica entre esses dispositivos, considerando uma perspectiva bidimensional e uma reta entre as coordenadas de ambos. Logo, a distância entre um servidor central e um cliente é definida pela distância euclidiana. Ou seja, considerando que a coordenada de um cliente é $P_c = (x_1, y_1)$ e de um servidor central é $P_s = (x_2, y_2)$, a distância entre c e s é definida por $d(P_c, P_s) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Uma vez conectado a um servidor (s_x), um cliente (c) migra de conexão (de s_x para s_y) se, e somente se, sai da área de cobertura de s_x, entra na área de cobertura de s_y e a distância $d(P_c, P_y)$ é menor do que qualquer distância entre o c e um outro servidor, ou seja $\forall w \in S, d(P_c, P_y) \leq d(P_c, P_w)$. Assim, uma migração de um cliente prioriza sempre o

servidor central mais próximo.

O mapa desse cenário foi elaborado com auxílio do OSMWebWizard considerando uma sub-região da cidade de Campina Grande, já apresentado na Seção 5.4. O mapa gerado pelo OSMWebWizard foi manualmente tratado para retirar pontos de falha e descontinuação. Essas falhas são oriundas do processamento de imagens do OSMWebWizard durante a identificação das vias de tráfego do mapa. O mapa resultante é apresentado através da Figura 6.16.

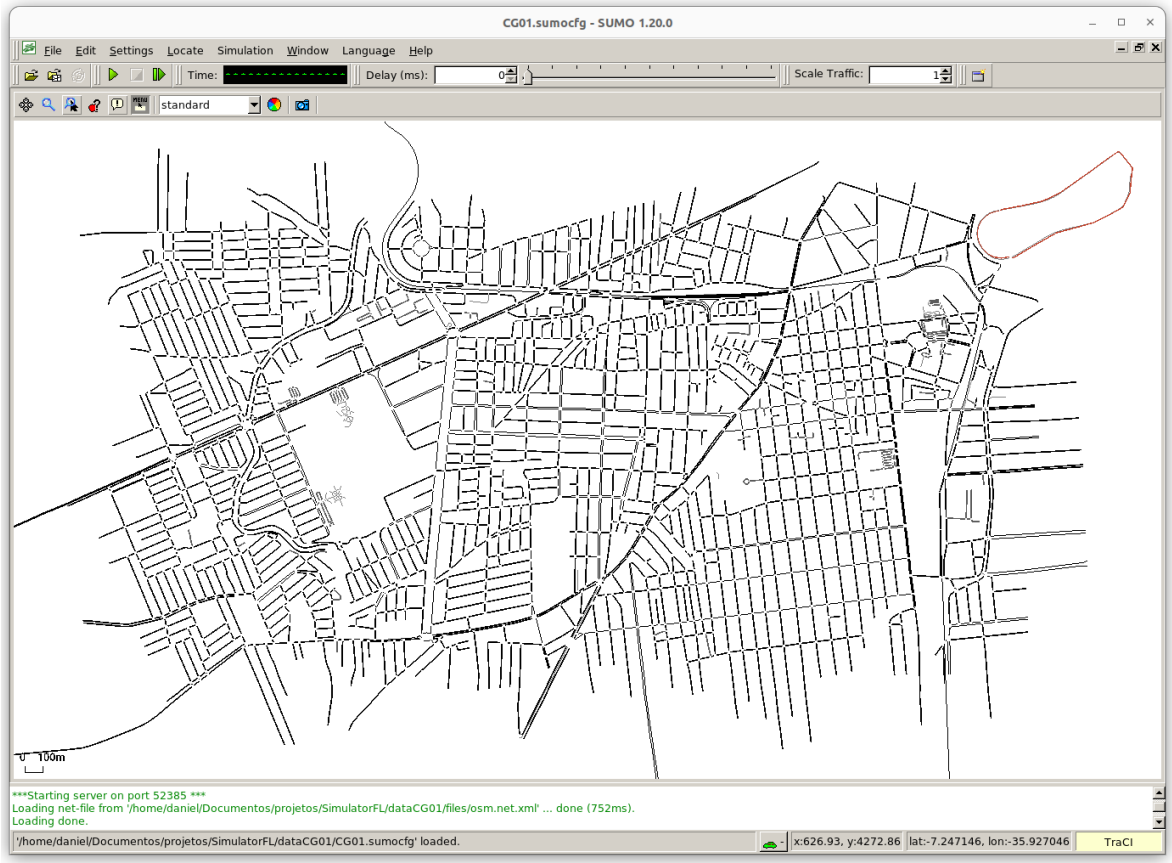


Figura 6.16: Mapa usado nas simulações do Cenário 4.

Especificamente nos cenários em que o algoritmo de coordenação utilizar a técnica de similaridade, o tamanho de lote definido para avaliação da similaridade é 32.

Sobre os servidores centrais, eles não apresentam mobilidade e, em todos os experimentos, são posicionados a partir do arranjo de uma matriz, conforme exemplificado na Figura 6.17, representando um cenário com 9 servidores arranjados em uma estrutura de grade, com 3 linhas e 3 colunas. Por sua vez, a posição inicial dos clientes, assim como a rota, é definida aleatoriamente pelo SUMO.

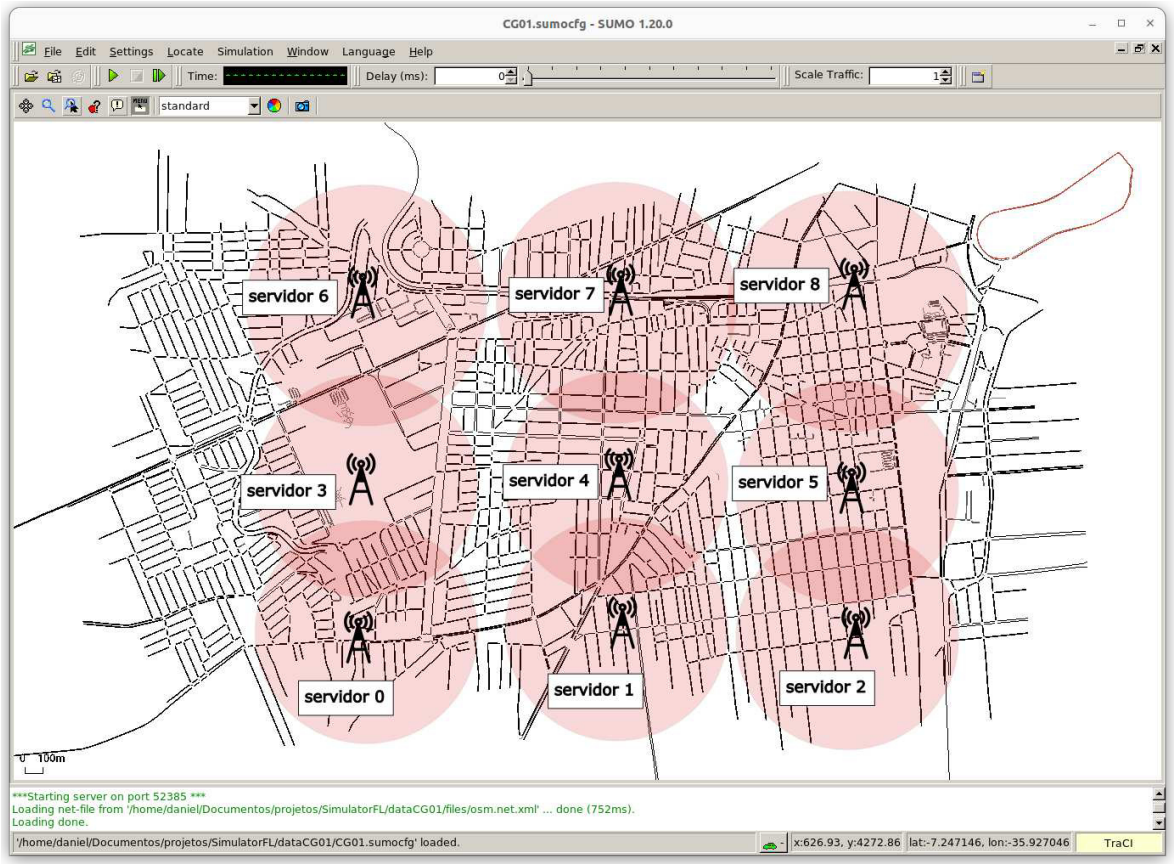


Figura 6.17: Representação do posicionamento aproximado dos servidores centrais no mapa do Cenário 4.

Ao todo 2756 clientes (veículos) foram criados durante a simulação, sendo 95 ônibus 672 motos e 1989 carros. O SUMO também definiu a rota e o instante de partida de cada cliente na simulação. Comparativamente aos Cenários 1, 2 e 3, houve um aumento significativo do número de clientes e de servidores de borda, aumentando o tempo de processamento de simulação, além do consumo de memória usada no FLSimulator. Nesse experimento, cada cliente possui uma memória capaz de armazenar 15000 amostras e a distribuição de amostras é executada considerando a distribuição de blocos de 2000 amostras, a cada 50 passos de simulação.

Por fim, assim como nos cenários anteriores, a simulação dura 2000 passos de simulação. Todavia, a escolha aleatória de um novo domínio de problema, simulando um desvio de aprendizagem, ocorre no passo 1250.

6.5.1 Resultados

Da mesma forma que nos cenários anteriores, inicia-se a análise dos experimentos através do gráfico da Figura 6.18, que demonstra a migração de clientes em cada servidor central ao longo do experimento.

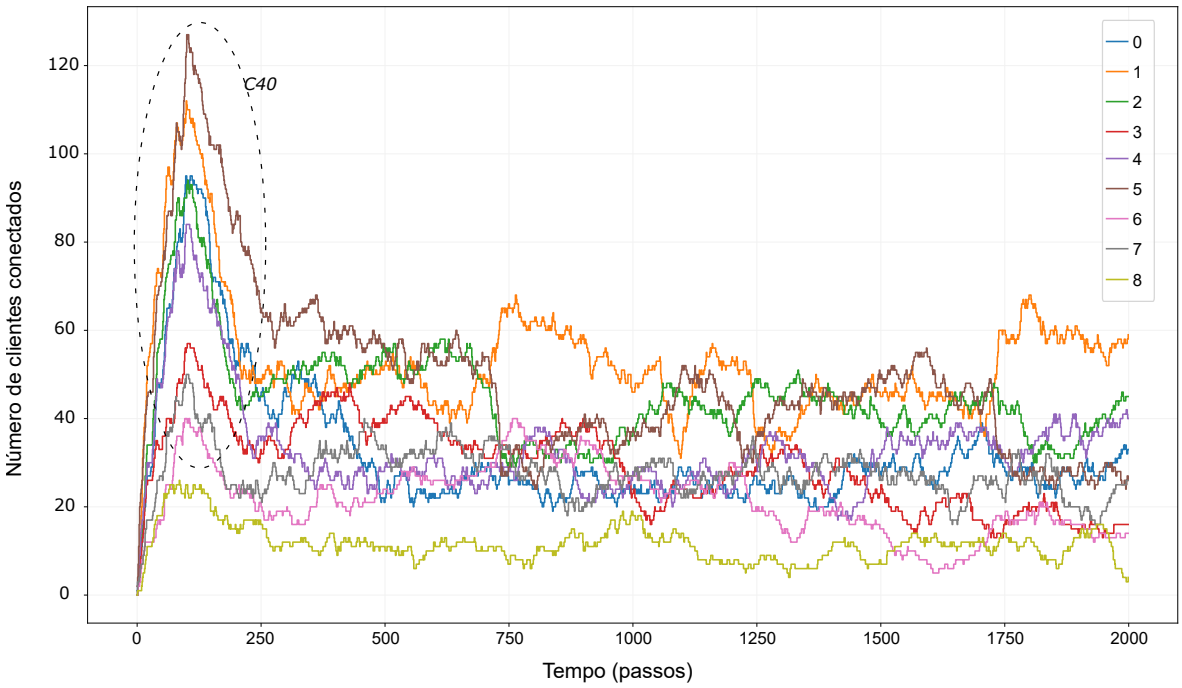


Figura 6.18: Gráfico de conexões de clientes para cada servidor central referente ao Cenário 4.

Assim como nos demais, conclui-se através da Figura 6.18 que houve migração de clientes durante todo experimento. Além disso destaca-se que os 2756 veículos foram criados em instantes distintos da simulação (pelo SUMO, como já mencionado).

Comparando a Figura 6.18 com os respectivos gráficos dos Cenários 1, 2 3, observa-se um comportamento de migração distinto. Nos cenários anteriores, a rota dos usuários eram cíclicas e não saiam da aplicação. Ou seja, qualquer usuário estava conectado a um domínio durante toda a simulação e o número total de clientes no cenário simulado permaneceu o mesmo. Além disso, considerando a topologia do grafo, os clientes podiam sair de um domínio de um servidor central e voltar ao mesmo domínio posteriormente.

Por sua vez, no Cenário 4, através da simulação de mobilidade pelo SUMO, os clientes podem ir para zonas mortas, ou seja, sem cobertura de nenhum servidor central, ficando sem domínio. Após a conclusão da rota, o cliente é extinto da simulação e, por isso, o

número total de clientes varia ao longo da simulação. Além disso, considerando a topologia da Figura 6.17, um cliente ao sair da área de cobertura de um servidor central dificilmente retorna para essa mesma cobertura. Isso é justificado pela representação da área de cobertura de um servidor central por um círculo e pelo fato de cada veículo ter apenas um destino.

Através da Figura 6.18 também observa-se que há um pico na quantidade de clientes conectados ao domínio no intervalo $[0 - 250]$, destacado por $C40$. Após esse intervalo, há uma queda no número de clientes, mantendo uma média de clientes conectados aos diferentes servidores centrais visivelmente menor que nos cenários anteriores. Enquanto nos Cenários 1, 2 e 3, em apenas 3 instantes ocorreu de algum servidor central manter menos de 50 clientes em seu domínio, no Cenário 4, apenas o $servidorcentral1$ manteve-se a maior a parte da simulação com mais de 50 clientes conectados (ou próximo a isso) ao seu domínio.

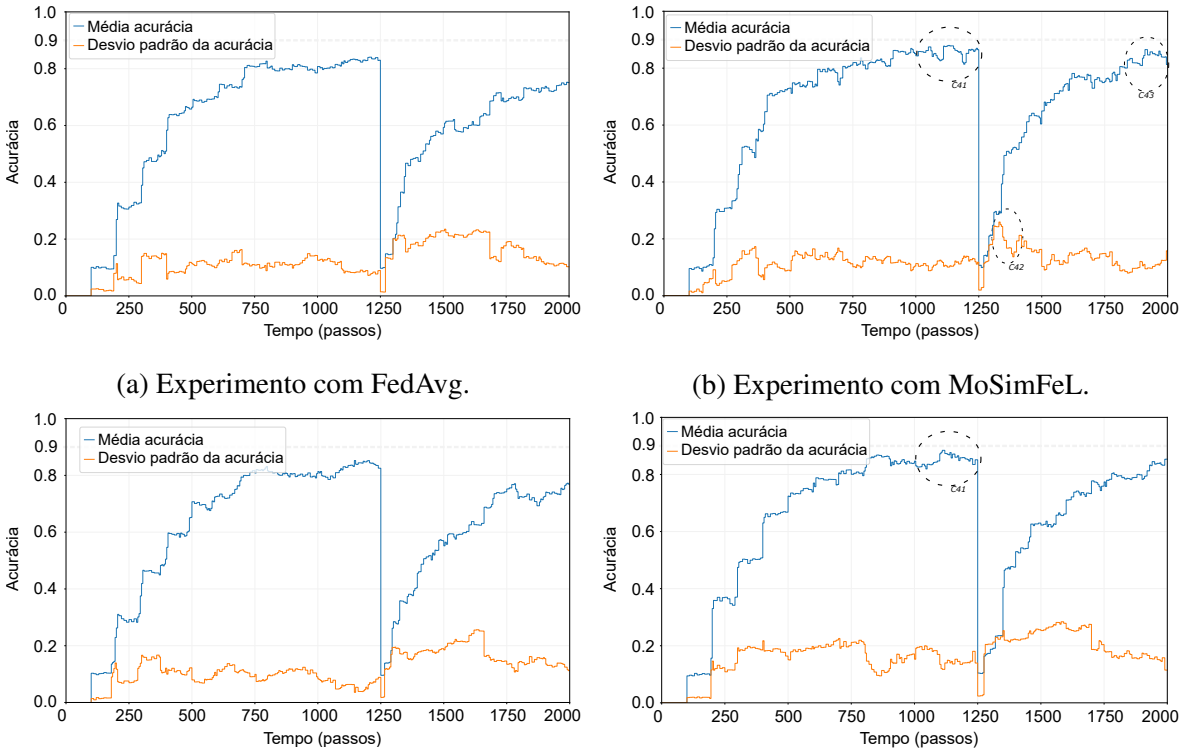
O comportamento de migração do Cenário 4 demonstra a dificuldade que os servidores centrais tiveram para executar o treinamento federado devido a alta mobilidade dos dispositivos e a diminuição da média do número de clientes para o número de servidores centrais.

No gráfico da Figura 6.19 é apresentado a média e o desvio padrão da avaliação da acurácia dos modelos ao longo do experimento.

Na Figura 6.19, é possível identificar que nenhum algoritmo conseguiu alcançar a média da acurácia mínima requisitada pela aplicação em nenhum instante. Todavia, os algoritmos MoSimFeL e SimFeL chegaram mais próximo desse feito, conforme destacado pelo marcador $C41$ nos gráficos das Figura 6.19b e 6.19d. Destaca-se que, nos gráficos da Figura 6.19, o valor $0,9$ foi destacado no eixo da acurácia para realçar a diferença entre a acurácia requisitada pela aplicação e acurácia alcançada.

Ainda sobre a Figura 6.19, identifica-se que o MoSimFeL obteve um menor desvio padrão, ultrapassando o limiar de 0.2 apenas por um curto período, destacado pelo marcador $C42$, enquanto os demais algoritmos ultrapassaram esse limiar por um intervalo maior. Assim, o algoritmo MoSimFeL conseguiu demonstrar uma curva da média da acurácia acima dos demais algoritmos e uma homogeneidade entre as acurácias maior do que os demais algoritmos.

A incapacidade de todos os algoritmos alcançarem o limiar de 0.9 demonstra como o Cenário 4 apresentou um cenário de mobilidade que dificultou mais o treinamento do que o Cenário 1. Todavia, na Figura 6.19, evidencia-se a robustez do MoSimFeL em se adaptar



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.19: Média e desvio padrão da acurácia dos modelos de todos os servidores centrais calculada pelos clientes referente ao Cenário 4.

a ambientes não estacionários, sendo o único algoritmo a alcançar o mesmo patamar de avaliação de acurácia após o modelos apresentarem desvio de aprendizagem, como destacado pelo marcador $C43$.

Na Tabela 6.5, exibe-se o número de ciclos de treinamento executados em cada cenário. É possível observar que as aplicações que não consideraram a mobilidade executaram um número menor de ciclos de treinamento do que os cenários com algoritmos que consideraram mobilidade. Esse comportamento foi similar aos cenários anteriores. Em relação ao MoSimFeL, o FedAvg executou $2,09\%$ menos treinamentos e o SimFeL executou $11,88\%$ menos treinamentos. Comparativamente ao MoFeL, o FedAvg executou $8,49\%$ menos treinamentos e o SimFeL executou $17,64\%$ menos treinamentos. Isso demonstra a superioridade do MoSimFeL e do MoFeL em executar treinamentos em contextos com mobilidade, mesmo apresentando abandono do número de treinamentos.

O número de abandonos de treinamento dos clientes nos algoritmos é justificado pelas falhas de previsibilidade da rota dos clientes calculadas pelo SUMO. As falhas ocorreram

Algoritmos	QA	QT
FedAvg	2504	140
MoSimFeL	1258	143
MoFeL	1491	153
SimFeL	2432	126

Tabela 6.5: Número de abandonos e ciclos concluídos referente ao Cenário 4.

devido a eventos simulados do tráfego de veículos, retardando a expectativa de chegada dos clientes nos pontos estipulados da rota em relação ao tempo inicialmente previsto.

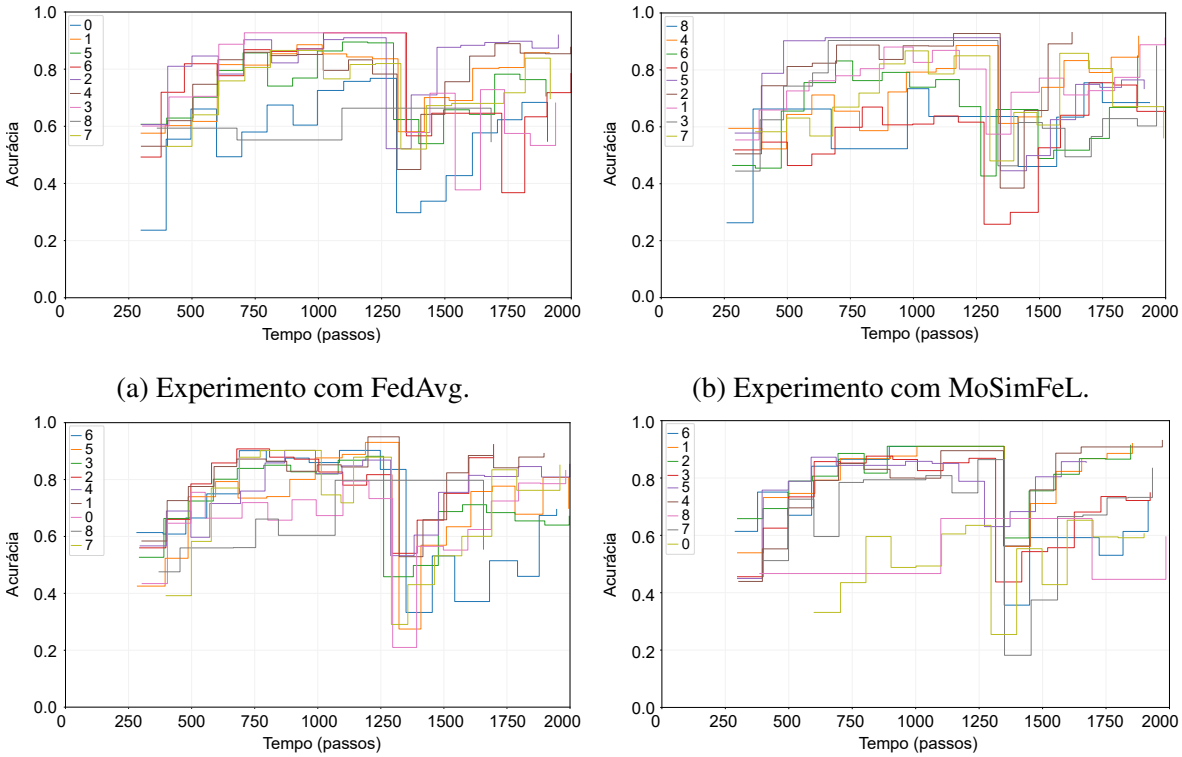
Diferentemente dos cenários anteriores, o SimFeL conseguiu concluir menos treinamentos que o FedAvg. A priori, essa informação invalidaria a conclusão dos cenários anteriores, que atestava a capacidade do SimFeL em avaliar indiretamente o tempo de chegada de um cliente em um domínio a partir da avaliação de sua base de dados. Todavia, no Cenário 4, o curto tempo de vida dos clientes na simulação, que na maioria dos casos migraram no máximo uma vez, não favoreceu a formação de bases de dados que pudessem ser avaliadas para analisar a origem dos dispositivos. Assim, na maioria dos casos, ao migrar para um novo domínio, um cliente permaneceu por pouco tempo na simulação até concluir sua rota. Dessa maneira, a conclusão anterior sobre a capacidade de avaliação indireta da capacidade de conclusão do treinamento nos cenários simulados mantém-se válida.

A análise da Tabela 6.5, conjuntamente aos gráficos da Figura 6.19, demonstra que o desempenho do MoSimFeL foi superior ao dos demais algoritmos, mesmo com um menor número de abandonos de treinamento. Isso sugere que, em vez de simplesmente aumentar o número de clientes selecionados na expectativa de que alguns concluam o treinamento, é mais eficiente analisar a base de dados e avaliar a disponibilidade de cada cliente, de modo a selecionar aqueles que efetivamente contribuirão para o treinamento.

Na Figura 6.20, apresenta-se os gráficos do valor de ACC_A pelo tempo. Nesses gráficos, cada curva é análise de ACC_A de cada servidor central.

Nas Figuras 6.22 e 6.21, apresenta-se respectivamente os gráficos de ACC_O e ACC_N pelo tempo para cada algoritmo avaliado.

Na análise dos gráficos apresentados nas Figuras 6.20, 6.22 e 6.21, é preciso reconhecer



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

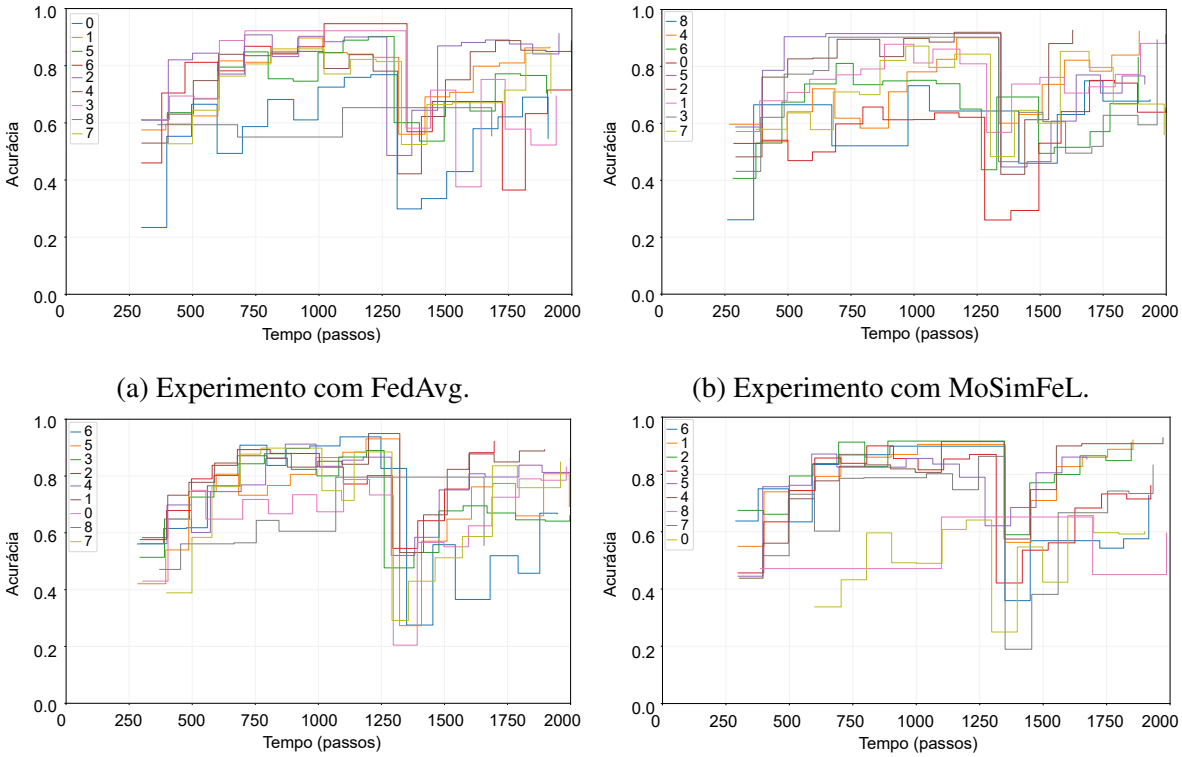
(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.20: ACC_A ao longo da simulação de todos os servidores centrais referente ao Cenário 4.

que a avaliação do treinamento a partir de uma perspectiva de clientes mais antigos ou mais recentes é prejudicada pelo tempo de vida dos clientes simulados via SUMO. Assim, poucos clientes conseguem alcançar um tempo de vida superior a dois ciclos de treinamento. Isso é facilmente perceptível no gráfico de ACC_O do MoSimFeL (Figura 6.22c), em que o servidor central 8 não conseguiu coletar ao menos um ponto para o gráfico. Ou seja, após todos os treinamentos, não houve um cliente que estivesse no domínio do servidor 8 e já tivesse participado de algum treinamento anteriormente.

Nesse sentido, os gráficos de ACC_A e ACC_N são semelhantes, visto que a maior parte dos clientes do domínio de um servidor não participou de nenhum treinamento anterior, devido ao tempo de vida dos clientes. A análise dos gráficos de ACC_A ratificam o desvio padrão apresentado nos gráficos da Figura 6.19. Os treinamentos do MoSimFeL e SimFeL apresentaram resultados mais homogêneos, repercutindo na avaliação de eficiência dos clientes.



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

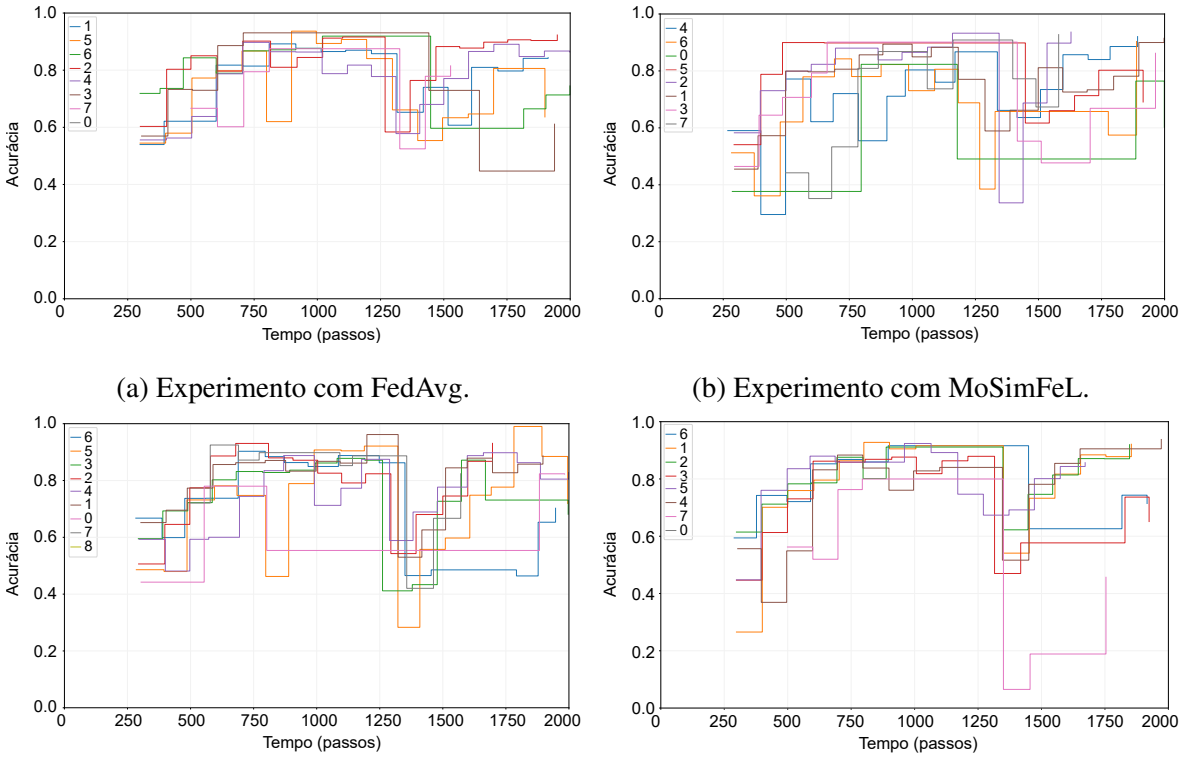
(d) Experimento com SimFeL.

Figura 6.21: ACC_N ao longo da simulação de todos os servidores centrais referente ao Cenário 4.

Diferentemente dos cenários anteriores, o MoFeL não conseguiu resultados tão satisfatórios e, mais do que isso, apresentou resultados menos satisfatórios que o SimFeL. Nesse sentido, a coordenação do MoFeL foi prejudicada pela imprecisão das rotas dos clientes a partir da posição geográfica. Dado o erro na predição das rotas dos clientes, é possível até mesmo conjecturar que a análise de mobilidade via base de dados, discutida nos cenários anteriores, pode inferir a disponibilidade dos clientes de forma mais precisa que a análise das rotas. Assim, o Cenário 4 valida a importância de analisar a similaridade na coordenação de AF em cenários com mobilidade.

6.6 Considerações Finais

Ao optar pelo treinamento de modelos distintos para cada servidor central, o MoSimFeL viabiliza a especialização de modelos a partir de diferentes domínios da aplicação. Nos experimentos apresentados, as diferenças entre os domínios são inerentes à posição dos cli-



(a) Experimento com FedAvg.

(b) Experimento com MoSimFeL.

(c) Experimento com MoFeL.

(d) Experimento com SimFeL.

Figura 6.22: ACC_O ao longo da simulação de todos os servidores centrais referente ao Cenário 4.

entes nos cenários simulados e, por isso, os servidores centrais são instalados em posições distintas. Caso contrário, a presença de mais de um servidor na mesma posição implicaria no treinamento de modelos parecidos, ou seja, com alta similaridade, por serem treinados dentro da perspectiva do mesmo domínio, não trazendo benefícios para a aplicação quanto à abordagem de multidomínios.

Apesar dos experimentos apresentados não abordarem outras perspectivas de cenários multidomínios, considerando aspectos de negócio da aplicação, por exemplo, a proposta do MoSimFeL não se limita ao tratamento de multidomínios por regionalização.

A análise de mobilidade dos dispositivos possibilitou a diminuição do abandono de clientes na fase de treinamento à medida que avalia, mesmo que de forma inexata, a perspectiva de um cliente se manter conectado ao domínio do servidor central durante um ciclo de treinamento. Mais do que isso, através da mobilidade, dependendo da estratégia adotada, o servidor central consegue prever a conexão de novos clientes ao seu domínio, para avaliar o

melhor instante para iniciar o ciclo de treinamento.

Apesar desta tese ter avaliado a influência do abandono de treinamento exclusivamente a partir da mobilidade, é possível estender a proposta para contemplar outros fatores, tais como falhas no canal de comunicação, falta de energia dos clientes provocando seu desligamento ou a realocação de recursos computacionais dos clientes para a execução de outras tarefas de maior prioridade que surgem inesperadamente. Para isso, assim como foi avaliada a mobilidade, é possível definir uma função que pondere a perspectiva de um cliente em participar efetivamente do treinamento local, considerando diferentes fatores, como os exemplificados.

O MoSimFeL também possibilitou contornar o problema do esquecimento catastrófico ao contemplar clientes com bases de dados que foram o foco de treinamentos anteriores. Ou seja, bases de dados similares às que já tinham sido usadas em diferentes instâncias dos modelos são usadas novamente durante o ensinamento de novas tarefas, com o intuito de propagar para as novas gerações dos modelos os ensinamentos antigos anteriormente assimilados. Nesse sentido, em todos os cenários simulados, o MoSimFeL mostrou-se mais eficiente quanto a análise de ACC_O comparativamente aos demais algoritmos.

Assim, comparado a outros algoritmos de coordenação de AF, o MoSimFeL possibilita prolongar o aprendizado das lições, aproveitando os esforços empenhados em cada treinamento. Apesar disso, a manutenção de lições antigas em novos aprendizados possui limites, devido à capacidade de aprendizado da arquitetura das redes profundas. Caso contrário, um modelo único seria capaz de aprender todos os domínios e poderia ser usado por toda a aplicação, invalidando a justificativa de uma estratégia de multidomínios em vários cenários, tais como os multidomínios por regionalização.

Nesse contexto, à medida que um determinado conjunto de amostras é reiteradamente desconsiderado a cada novo treinamento, as nuances específicas que foram apreendidas graças ao conjunto de dados serão esquecidas. Nesse sentido, o esquecimento também é importante para a qualidade do modelo, pois o esquecimento de tarefas que não são mais úteis possibilita que o modelo aprenda novas lições. Esse comportamento também é natural do ser humano que, ao longo da vida, esquece de aprendizados que não foram significativos ou que não são reforçados por um longo período de sua vida. Portanto, é importante enfatizar que o algoritmo possibilita prolongar a manutenção de uma lição no modelo e não a mantém indefinidamente.

A avaliação de similaridade também pode ser usada em outras técnicas de treinamento tradicionais distribuídas e sequenciais a fim de otimizar o treinamento. No treinamento distribuído tradicional, em que um modelo é treinado por diferentes núcleos de processamento, é vantajoso distribuir subconjuntos de dados para cada núcleo, considerando a similaridade entre si. Dessa maneira, cada núcleo treina um conjunto de dados com amostras mais próximas entre si, acelerando a convergência do treinamento.

O treinamento contínuo tradicional, em aplicações que não utilizam AF, também pode ser beneficiado com a análise de similaridade. Nesse caso, considerando que não há restrição quanto ao acesso de dados pela aplicação, é possível analisar a similaridade entre os subconjuntos disponíveis para o retreinamento e o conjunto de dados usado em treinamentos anteriores. Dessa forma, é possível identificar quais subconjuntos contribuirão de forma mais significativa para o retreinamento em comparação com os outros subconjuntos. Com a diferenciação entre os subconjuntos, o retreinamento pode diminuir o volume de dados utilizados, considerando apenas os subconjuntos que mais contribuirão para o treinamento, empregando menos recursos computacionais e acelerando a convergência do treinamento.

Todas as perspectivas do uso de similaridade em outras técnicas de treinamento em AM permeiam por uma estratégia: prever o efeito de um conjunto de dados (D_1) no retreinamento de um modelo a partir da análise similaridade direta entre D_1 e outros conjuntos já usados em treinamento D_X já usados em treinamentos anteriores desse domínio. Ainda é possível estender essa estratégia para que seja aplicada no primeiro treinamento de um modelo, ao analisar a similaridade entre as amostras de um mesmo conjunto de dados e eliminar amostras redundantes, ou seja, com alta similaridade. É previsível que essa análise exija grandes esforços computacionais para a análise e que poderiam ser direcionados para o treinamento considerando toda a base de dados, tornando a avaliação da similaridade injustificada.

Todavia, é possível aplicar o mesmo método para avaliar a similaridade entre subconjuntos de dados, diminuindo os esforços computacionais para avaliar a similaridade, em detrimento de uma análise menos precisa, à medida que analisa dados coletivamente ao invés de individualmente. Nesse sentido, o consumo de recursos computacionais na análise de similaridade pode justificar a melhoria de performance no treinamento do modelo.

Para o MoSimFeL também foi abordada a avaliação do desvio de conceito com uma constante avaliação do modelo mediante requisitos da aplicação. Para isso, solicita-se que os

clientes avaliem os modelos individualmente. Essa avaliação pode ser um ponto de fragilidade do requisito de privacidade de dados, que é o princípio norteador da AF, pois é possível inferir a base de dados do cliente a partir da percepção do modelo. Isto é, um agente malicioso pode interceptar essa avaliação ao ser transmitida entre o cliente e o servidor central, para inferir os dados do cliente.

Para contornar essa fragilidade, é possível antecipar a avaliação dos clientes também com base na similaridade. Nesse caso, a similaridade entre instâncias de modelos dos quais um cliente participou do treinamento e o modelo a ser avaliado pode inferir a satisfação do cliente em relação ao modelo a ser avaliado. Essa hipótese é respaldada a partir da premissa de que, se o cliente participou do treinamento de um modelo, o modelo considerou a base de dados desse cliente em seu aprendizado e correspondeu às perspectivas de aprendizado desse cliente.

Essa premissa é factível quando os treinamentos dos modelos, dos quais o cliente participou, foram eficientes. Assim, caso esses modelos históricos tenham uma baixa similaridade com o modelo avaliado, é possível inferir que o modelo avaliado não atenderá às demandas desse cliente. Em contrapartida, caso haja uma alta similaridade, o modelo avaliado atenderá às demandas desse cliente da mesma forma que os modelos anteriores atenderam.

Ou seja, através dessa estratégia, não é necessário que o cliente compartilhe a sua percepção sobre a eficiência do modelo e é possível antecipar indiretamente a sua avaliação do modelo. Evidentemente, a estratégia não é precisa e cabe aos requisitos da avaliação ponderarem sobre aumentar a segurança na confidencialidade dos dados dos clientes em detrimento de erros quanto a inferência de satisfação do cliente.

Por fim, é importante enfatizar que as propostas listadas são intuitivamente coerentes. Todavia, elas são dispostas neste trabalho como hipóteses visto que não foram avaliadas precisamente para comprovação. Se comprovadas, até onde foi alcançado na revisão da literatura, essas estratégias são inovadoras. Até então, as estratégias conhecidas e disponíveis na literatura enumeram boas práticas para a seleção de dados para o treinamento dos modelos e não conseguem quantificar antecipadamente a influência dos dados no treinamento.

A antecipação da influência (por mais que inexata) de uma base de dados no treinamento de um modelo é diretamente relacionada ao investimento de recursos computacionais necessários para o treinamento do modelo AM. Considerando o custo para alocação desses

recursos e o treinamento de complexos modelos com grandes bases de dados cada vez mais empenhados em aplicações reais, a avaliação antecipada das bases de dados pode resultar em economias financeiras significativas no treinamento. Além disso, é possível que o aumento da performance do treinamento viabilize aplicações que exigem um tempo máximo para retreinamento dos seus modelos.

Por fim, conclui-se que o MoSimFeL possibilitou que satisfazer o objetivo em fomentar o treinamento de modelos de AM em aplicações de AF sob o contexto de multidomínios, não-estacionários e com mobilidade.

Capítulo 7

Conclusão

A Aprendizagem de Máquina tem sido uma importante ferramenta no desenvolvimento de aplicações com IA. Todavia, à medida que cresce a demanda pelo uso de AM em diferentes aplicações, também cresce a preocupação com os dados usados para o treinamento de modelos que atendem a essas aplicações. Nesse contexto, surge a Aprendizagem Federada (AF).

A proposta principal desta tese foi investigar a mobilidade, a não-estacionariedade e os multidomínios na AF. Reconhecendo a necessidade de desenvolver estratégias para garantir o treinamento federado nesse contexto, foi proposto o algoritmo MoSimFeL, que é um novo algoritmo de coordenação de treinamento federado, inspirado no FedAvg, e que atende a aplicações com essas características. Para isso, são usadas duas informações que são desconsideradas para o FedAvg: a mobilidade dos usuários e o histórico de participação de clientes.

Os resultados dos experimentos apresentados no Capítulo 6 demonstraram que o MoSimFeL alcançou resultados de treinamento mais eficientes em todos os cenários, destacando-se principalmente no contexto que o motivou. Todavia, a avaliação de similaridade requer considerações de esforços computacionais que não foram precisamente mensurados. Nesse sentido, pretende-se, em trabalhos futuros, avaliar o tempo empenhado na análise do MoSimFeL e propor melhorias para acelerar a execução do algoritmo.

Além das contribuições para a AF, o trabalho apresentado nesta tese elucidou novos desafios a serem investigados. Nas próximas seções, alguns desses desafios são abordados, compondo um conjunto de propostas para trabalhos futuros.

7.1 AF e a Não-Estacionariedade

O MoSimFeL abordou o desafio do esquecimento catastrófico, avaliando indiretamente a base de dados dos clientes a partir do seu histórico de treinamento. Com isso, o treinamento conseguiu priorizar novas bases de dados e aprender lições relacionadas a elas, ao mesmo tempo em que mantinha amostras já contempladas em treinamentos anteriores para preservar lições passadas.

No entanto, à medida que o domínio muda drasticamente, o MoSimFeL tem dificuldade de atualizar o modelo, mesmo que, comparativamente aos outros algoritmos, apresente melhores resultados.

Sobre isso, é possível pensar em melhorias para trabalhos futuros:

1. Aumentar o número de domínios do problema com a perspectiva de especializar o modelo, viabilizando a conexão do cliente ao modelo mais adequado.

2. Aumentar a capacidade de aprendizagem de todos os modelos para propiciar que assimilem novos aprendizados.

O aumento de domínios implica no aumento número de servidores centrais e de registros para guardar o histórico de modelos treinados. Para isso, há necessidade de mais memória para esses registros. Considerando que o servidor pode ser alocado em um servidor de borda, a memória não deve ser um empecilho. Todavia, o aumento do número de domínios requer mais treinamentos para manutenção de todos os modelos, incluindo o retreinamento quando necessário, repercutindo em esforço computacional significativo para os clientes.

Nesse sentido é possível levantar as questões:

1. Como definir o número de domínios de uma aplicação considerando os requisitos de performance do modelo e extrapolando os quesitos de negócio da aplicação e de infraestrutura?

2. Como alterar dinamicamente os domínios a fim de conquistar a melhor relação custo-benefício entre o consumo de recursos computacionais para manter os domínios e a confiabilidade em atender aos requisitos da aplicação?

A primeira questão acima extrapola a arquitetura proposta nesta tese. É possível implementar mais de um servidor central (com seu respectivo domínio) em um servidor de borda ou até mesmo mais de um domínio para o mesmo escopo do modelo de negócio da aplicação. No exemplo do Capítulo 3, mencionou-se o exemplo de implementação de um domínio para cada comércio virtual em uma aplicação de avaliação do perfil consumista de clientes. Estendendo esse exemplo, seria possível a implementação de mais domínios por comércio virtual. Nesse caso, cada domínio se especializaria em um subproblema. Em relação à restrição de infraestrutura, foi proposta uma arquitetura na qual cada servidor central é implementado em um servidor de borda, conforme Figura 3.1, e o domínio do servidor central abrange todos os clientes que estão conectados especificamente à sua borda. No entanto, expandindo a arquitetura proposta, é possível que vários servidores centrais sejam implementados no mesmo servidor de borda e compartilhem o mesmo conjunto de clientes, particionando em vários subproblemas.

Em relação à segunda questão, é preciso destacar que manter muitos domínios desnecessariamente representa um desperdício de recursos computacionais. Por outro lado, manter menos domínios do que o necessário implica em domínios incapazes de abranger todos os aspectos do problema e de treinar modelos capazes de assimilar todos os aprendizados. Esta tese não abordou um método inteligente para definir quando e como serão estabelecidos os domínios dos servidores centrais, argumentando que isso é determinado exclusivamente pela aplicação. Nesse sentido, a aplicação deve conhecer os domínios do seu problema para projetar a estratégia de treinamento. No entanto, o dinamismo pode gerar situações inesperadas, sendo fundamental um mecanismo automatizado para reconhecer essas transformações. Assim, projeta-se, em trabalhos futuros, o estudo e desenvolvimento de técnicas para decidir de forma otimizada como implementar os domínios na aplicação.

Outro ponto de investigação a ser aprofundado é a análise da previsão da contribuição de um cliente a partir do treinamento local, considerando todo o histórico de treinamento do cliente. Na versão do MoSimFeL apresentada, considerou-se apenas o último treinamento executado pelo cliente. Essa escolha foi amparada pela não-estacionariedade do sistema, de modo que os treinamentos mais recentes foram realizados em contextos mais atuais do sistema, tornando a base de dados desse treinamento mais similar à situação atual do cliente do que as bases usadas em treinamentos anteriores. Além disso, embora não tenha sido de-

monstrado matematicamente ou experimentalmente, é evidente que a análise de similaridade é uma operação complexa que requer recursos computacionais, sendo um fator limitante na sua adoção. Assim, a comparação de vários níveis de similaridade a partir do histórico completo do cliente, avaliando diferentes treinamentos feitos por ele, pode ser inviável dependendo dos recursos da aplicação.

Todavia, uma análise completa pode aumentar a precisão na tomada de decisão e tornar-se imprescindível para aplicações que limitam o número de treinamentos. Nesses casos, cada ciclo de treinamento teria um desempenho ainda mais elevado, facilitando um aprendizado eficaz.

Outra vantagem de analisar o histórico completo de treinamentos é entender como a base de dados do cliente está variando, respeitando a privacidade prevista na AF. Compreender o dinamismo da base de dados do cliente permite antecipar as mudanças e iniciar o treinamento para identificar precocemente desvios de aprendizagem no modelo, possibilitando o retreinamento preventivo em vez de corretivo. Assim, o retreinamento deixaria de ser uma reação ao desvio do modelo e passaria a ser uma medida preventiva, garantindo a eficiência do modelo ao longo de seu ciclo de vida.

Por fim, propõe-se avaliar outras técnicas de comparação da similaridade, além das propostas nas Equações 2.8, 2.7, 2.6 e 2.5. Essas técnicas são necessárias na análise de similaridade do MoSimFeL, especificamente nas linhas [10 - 12] do algoritmo 27. Todavia, essas equações visam calcular a similaridade através de um escalar, o que pode omitir informações importantes da estrutura do modelo treinado. Nesse sentido, espera-se que a avaliação da similaridade através de uma estrutura multidimensional permita comparar a similaridade entre modelos não apenas pela distância escalar, mas também pelo posicionamento espacial das similaridades, através de um espaço multidimensional. Para isso, algoritmos como K-means podem agrupar as análises de similaridade, permitindo contemplar a seleção de clientes em cada grupo e, consequentemente, garantindo uma maior diversidade dos dados [24].

7.2 AF e Computação de Borda e Nuvem

No MoSimFeL, se um cliente migrar de domínio durante a execução de um treinamento local, o treinamento é interrompido e o empenho computacional investido na execução parcial

é perdido, mesmo que isso ocorra um instante antes da conclusão. Isso é frustrante para a aplicação. A priori, é possível propor que o treinamento seja concluído em outro domínio e o resultado seja enviado para o servidor central que solicitou o treinamento, mesmo que essa comunicação precise sair da borda, ir para a nuvem e chegar à borda desse servidor.

No entanto, do lado do cliente que migrou de domínio, sua contribuição será irrelevante, pois já está participando de outro domínio e utilizará outro modelo. Assim, para o cliente, mesmo que próximo à conclusão, é mais adequado abandonar o treinamento e assumir o prejuízo computacional parcial do que concluir o treinamento e empenhar mais recursos para um modelo do qual não irá usufruir. Por outro lado, na perspectiva da aplicação, é importante que o cliente conclua o treinamento independentemente de usufruir do modelo.

Assim, é fundamental quantificar o custo de conclusão do treinamento em uma borda distinta para avaliar de forma precisa a decisão sobre a conclusão ou interrupção do treinamento. Futuramente, espera-se realizar um estudo para mensurar de forma precisa a quantificação desse custo.

7.3 Transferência de Aprendizado

Outra possibilidade seria investigar como a participação de treinamento local de um cliente pode ser aproveitada em outro domínio. O treinamento local de um modelo pode ser didaticamente representado pela operação $M_{x,1}\theta\nabla_{c,1} \rightarrow M_{y,1}$, em que $M_{x,1}$ é o modelo de entrada a ser retreinado, θ é operação de treinamento, $\nabla_{c,1}$ é o gradiente definido a partir da base de dados do cliente e $M_{y,1}$ é o modelo local definido após o treinamento. Se $M_{x,1}$ muda em decorrência da migração, o fator $\nabla_{c,1}$ perde o sentido do contexto e, consequentemente, $M_{y,1}$ também. Nesse contexto, surge a pergunta: *Como aproveitar* $\nabla_{c,1}$ *e* $M_{y,1}$ *no treinamento* $M_{x,2}\theta\nabla_{c,1} \rightarrow M_{y,2}$*?*

A partir dessa questão de investigação ampla, é possível definir outra questão mais específica: considerando que os fatores $M_{x,1}$, $M_{x,2}$, $M_{y,1}$ e $\nabla_{c,1}$ são conhecidos pelos clientes, há alguma relação deduzível entre $\nabla_{c,1}$ e $\nabla_{c,2}$ ou $M_{y,1}$ e $M_{y,2}$?

Essa questão de pesquisa se insere na área de transferência de aprendizado, pois envolve a transferência do conhecimento adquirido durante o treinamento local em um domínio para outro. Isso permite a otimização dos recursos computacionais utilizados pelo cliente, ao

considerar seu treinamento em um novo domínio, mesmo que tenha sido iniciado em outro. Pressupõe-se que o custo computacional na transferência deve ser menor do que o de um novo treinamento local e, caso contrário, é melhor que o cliente execute um novo treinamento no novo domínio. Todavia, isso não é necessariamente verdade, visto que, além de implicar na otimização de recursos, a transferência pode beneficiar o treinamento do modelo $M_{y,2}$ com o aprendizado trazido de outro domínio.

Para colaborar com a investigação desta questão, é possível considerar a similaridade entre os modelos de entrada. Conhecendo a similaridade entre $M_{x,1}$ e $M_{x,2}$, seria possível inferir que há a mesma relação entre $\nabla_{c,1}$ e $\nabla_{c,2}$? Consequentemente, dada essa relação e $\nabla_{c,1}$, seria possível definir $\nabla_{c,2}$? $\nabla_{c,2}$ é o fator determinante para a definição de $M_{y,2}$.

Também é possível investigar o resultado do treinamento de outros clientes e a similaridade entre os modelos locais dos clientes *nativos* com o *imigrante*. Nesse caso, a investigação partiria do questionamento: se em um cliente (cn) nativo ocorreu publicamente a operação $M_{x,2}\theta\nabla_{cn,1} \rightarrow M_{yn,2}$ e $M_{x,1}$ e $\nabla_{c,1}$ são conhecidos, é possível usar a relação de similaridade entre $M_{x,1}$ e $M_{x,2}$ e a relação entre $\nabla_{c,1}$ e $\nabla_{c,2}$ para definir $M_{y,2}$?

Claramente, os questionamentos apresentados são intrigantes e precisam ser analisados matematicamente e experimentalmente para determinar se este é um bom caminho para se aprofundar na proposição de uma solução. Confirmando as hipóteses levantadas, esse método pode efetivamente melhorar a eficiência dos treinamentos federados. O treinamento tradicional contínuo também poderia ser beneficiado mediante o aproveitamento do aprendizado por diferentes bases de dados em outros contextos, avançando o aprendizado de um modelo.

Bibliografia

[1] M Salehi Heydar Abad, Emre Ozfatura, Deniz Gunduz, and Ozgur Ercetin. Hierarchical federated learning across heterogeneous cellular networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8866–8870. IEEE, 2020.

[2] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.05.008. URL https://www.sciencedirect.com/science/article/pii/S1566253521001081.

[3] Karim Abouelmehdi, Abderrahim Beni-Hssane, Hayat Khaloufi, and Mostafa Saadi. Big data security and privacy in healthcare: A review. *Procedia Computer Science*, 113:73–80, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.08.292. URL https://www.sciencedirect.com/science/article/pii/S1877050917317015. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops.

[4] Karim Abouelmehdi, Abderrahim Beni-Hessane, and Hayat Khaloufi. Big healthcare data: preserving security and privacy. *Journal of big data*, 5(1):1–18, 2018.

[5] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du,

Ihsan Ali, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *IEEE Communications Surveys & Tutorials*, 22(3): 1646–1685, 2020. doi: 10.1109/COMST.2020.2988293.

[6] Hamdan O Alanazi, Abdul Hanan Abdullah, and Kashif Naseer Qureshi. A critical review for developing accurate and dynamic predictive models using machine learning methods in medicine and health care. *Journal of medical systems*, 41:1–10, 2017.

[7] Mohammed Aledhari, Rehma Razzak, Reza M. Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020. doi: 10.1109/ACCESS.2020.3013541.

[8] P. Ambika. Chapter thirteen - machine learning and deep learning algorithms on the industrial internet of things (iiot). In Pethuru Raj and Preetha Evangeline, editors, *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in Computers*, pages 321–338. Elsevier, 2020. doi: https://doi.org/10.1016/bs.adcom.2019.10.007. URL https://www.sciencedirect.com/science/article/pii/S0065245819300609.

[9] Sheraz Aslam, Herodotos Herodotou, Syed Muhammad Mohsin, Nadeem Javaid, Nouman Ashraf, and Shahzad Aslam. A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids. *Renewable and Sustainable Energy Reviews*, 144:110992, 2021. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2021.110992. URL https://www.sciencedirect.com/science/article/pii/S1364032121002847.

[10] Salvador V. Balkus, Honggang Wang, Brian D. Cornet, Chinmay Mahabal, Hieu Ngo, and Hua Fang. A survey of collaborative machine learning using 5g vehicular communications. *IEEE Communications Surveys & Tutorials*, 24(2):1280–1303, 2022. doi: 10.1109/COMST.2022.3149714.

[11] Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal K. Das, Luca Foschini, and Alessandro Zanni. A survey on fog computing for the internet of things. *Pervasive and Mobile Computing*, 52:71–99, 2019. ISSN 1574-1192. doi: https:

//doi.org/10.1016/j.pmcj.2018.12.007. URL https://www.sciencedirect.com/science/article/pii/S1574119218301111.

[12] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[13] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[14] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[15] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[16] Bouziane Brik, Adlen Ksentini, and Maha Bouaziz. Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems. *IEEE Access*, 8:53841–53849, 2020. doi: 10.1109/ACCESS.2020.2981430.

[17] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[18] Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. Knapsack problems — an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, 2022. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2021.

105693. URL https://www.sciencedirect.com/science/article/pii/S0305054821003889.

[19] Giuseppe Canonaco, Alex Bergamasco, Alessio Mongelluzzo, and Manuel Roveri. Adaptive federated learning in presence of concept drift. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021. doi: 10.1109/IJCNN52387.2021.9533710.

[20] Fernando E. Casado, Dylan Lema, Marcos F. Criado, Roberto Iglesias, Carlos V. Regueiro, and Senén Barro. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools Appl.*, 81(3):3397–3419, jan 2022. ISSN 1380-7501. doi: 10.1007/s11042-021-11219-x. URL https://doi.org/10.1007/s11042-021-11219-x.

[21] Theo Chow, Usman Raza, Ioannis Mavromatis, and Aftab Khan. Flare: Detection and mitigation of concept drift for federated learning based iot deployments. In *2023 International Wireless Communications and Mobile Computing (IWCMC)*, pages 989–995, 2023. doi: 10.1109/IWCMC58020.2023.10182870.

[22] Raffaele Cioffi, Marta Travaglioni, Giuseppina Piscitelli, Antonella Petrillo, and Fabio De Felice. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability*, 12(2), 2020. ISSN 2071-1050. doi: 10.3390/su12020492. URL https://www.mdpi.com/2071-1050/12/2/492.

[23] Laizhong Cui, Shu Yang, Fei Chen, Zhong Ming, Nan Lu, and Jing Qin. A survey on application of machine learning for internet of things. *International Journal of Machine Learning and Cybernetics*, 9(8):1399–1417, 2018.

[24] Laizhong Cui, Xiaoxin Su, Yipeng Zhou, and Lei Zhang. Clustergrad: Adaptive gradient compression by clustering in federated learning. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–7, 2020. doi: 10.1109/GLOBECOM42002.2020.9322527.

[25] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27:1071–1092, 2020.

[26] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[27] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[28] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[29] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25 (10):2283–2301, 2013. doi: 10.1109/TKDE.2012.136.

[30] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4): 12–25, 2015.

[31] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 228–237, 2021. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00042.

[32] S. Durga, Rishabh Nag, and Esther Daniel. Survey on machine learning and deep learning algorithms used in internet of things (iot) healthcare. In *2019 3rd Internati-*

onal Conference on Computing Methodologies and Communication (ICCMC), pages 1018–1022, 2019. doi: 10.1109/ICCMC.2019.8819806.

[33] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[34] Chenyuan Feng, Howard H. Yang, Deshun Hu, Zhiwei Zhao, Tony Q. S. Quek, and Geyong Min. Mobility-aware cluster federated learning in hierarchical wireless networks. *IEEE Transactions on Wireless Communications*, pages 1–1, 2022. doi: 10.1109/TWC.2022.3166386.

[35] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55 (5):94–100, 2017. doi: 10.1109/MCOM.2017.1600951.

[36] B. Ganguly and V. Aggarwal. Online federated learning via non-stationary detection and adaptation amidst concept drift. *IEEE/ACM Transactions on Networking*, (01): 1–11, jul 5555. ISSN 1558-2566. doi: 10.1109/TNET.2023.3294366.

[37] Bhargav Ganguly, Seyyedali Hosseinalipour, Kwang Taik Kim, Christopher G. Brinton, Vaneet Aggarwal, David J. Love, and Mung Chiang. Multi-edge server-assisted dynamic federated learning with an optimized floating aggregation point. *IEEE/ACM Transactions on Networking*, pages 1–16, 2023. doi: 10.1109/TNET.2023.3262482.

[38] Safwan M Ghaleb, Shamala Subramaniam, Zuriati Ahmed Zukarnain, and Abdullah Muhammed. Mobility management for iot: a survey. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):1–25, 2016.

[39] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Algorithmic Learning Theory*, pages 63–77, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31696-1.

[40] Dong-Jun Han, Minseok Choi, Jungwuk Park, and Jaekyun Moon. Fedmes: Speeding up federated learning with multiple edge servers. *IEEE Journal on Selected Areas in Communications*, 39(12):3870–3885, 2021. doi: 10.1109/JSAC.2021.3118422.

[41] Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G Flores, Jan Kautz, Daguang Xu, and Holger R. Roth. Do gradient inversion attacks make federated learning unsafe? *IEEE Transactions on Medical Imaging*, 42(7):2044–2056, 2023. doi: 10.1109/TMI.2023.3239391.

[42] J Matthew Helm, Andrew M Swiergosz, Heather S Haeberle, Jaret M Karnuta, Jonathan L Schaffer, Viktor E Krebs, Andrew I Spitzer, and Prem N Ramkumar. Machine learning and artificial intelligence: definitions, applications, and future directions. *Current reviews in musculoskeletal medicine*, 13:69–76, 2020.

[43] Aicha Idriss Hentati, Lobna Krichen, Mohamed Fourati, and Lamia Chaari Fourati. Simulation tools, environments and frameworks for uav systems performance analysis. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1495–1500, 2018. doi: 10.1109/IWCMC.2018.8450505.

[44] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one*, 15(4):e0230706, 2020.

[45] Tiansheng Huang, Weiwei Lin, Li Shen, Keqin Li, and Albert Y. Zomaya. Stochastic client selection for federated learning with volatile clients. *IEEE Internet of Things Journal*, pages 1–1, 2022. doi: 10.1109/JIOT.2022.3172113.

[46] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7865–7873, May 2021. doi: 10.1609/aaai.v35i9.16960. URL https://ojs.aaai.org/index.php/AAAI/article/view/16960.

[47] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Towards understanding biased client selection in federated learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10351–10375. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/jee-cho22a.html.

[48] Ji Chu Jiang, Burak Kantarci, Sema Oktug, and Tolga Soyata. Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, 20(21):6230, 2020.

[49] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[50] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*, pages 5834–5853. PMLR, 2023.

[51] Myeongkyun Kang, Soopil Kim, Kyong Hwan Jin, Ehsan Adeli, Kilian M. Pohl, and Sang Hyun Park. Fednn: Federated learning on concept drift data using weight and adaptive group normalizations. *Pattern Recognition*, 149:110230, 2024. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2023.110230. URL https://www.sciencedirect.com/science/article/pii/S0031320323009275.

[52] Hans Kellerer, Ulrich Pferschy, David Pisinger, Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Multidimensional knapsack problems*. Springer, 2004.

[53] Latif U. Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021. doi: 10.1109/COMST.2021.3090430.

[54] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97: 219–235, 2019. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2019.02.050. URL https://www.sciencedirect.com/science/article/pii/S0167739X18319903.

[55] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–

3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

[56] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[57] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2021. doi: 10.1109/TITS.2019.2962338.

[58] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A survey on security for mobile devices. *IEEE Communications Surveys & Tutorials*, 15(1):446–471, 2013. doi: 10.1109/SURV.2012.013012.00028.

[59] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[60] Chunlin Li, Yong Zhang, and Youlong Luo. A federated learning-based edge caching approach for mobile edge computing-enabled intelligent connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[61] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020.

[62] Shancang Li, Li Da Xu, and Shanshan Zhao. 5g internet of things: A survey. *Journal of Industrial Information Integration*, 10:1–9, 2018. ISSN 2452-414X. doi: https://doi.org/10.1016/j.jii.2018.01.005. URL https://www.sciencedirect.com/science/article/pii/S2452414X18300037.

[63] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37 (3):50–60, 2020. doi: 10.1109/MSP.2020.2975749.

[64] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks.

In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

[65] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020. doi: 10.1109/COMST.2020.2986024.

[66] Jessica Chia Liu, Jack Goetz, Srijan Sen, and Ambuj Tewari. Learning from others without sacrificing privacy: Simulation comparing centralized and federated machine learning on mobile health data. *JMIR mHealth and uHealth*, 9(3):e23728, 2021.

[67] Jianchun Liu, Hongli Xu, Lun Wang, Yang Xu, Chen Qian, Jinyang Huang, and He Huang. Adaptive asynchronous federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*, pages 1–1, 2021. doi: 10.1109/TMC.2021.3096846.

[68] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, pages 1–24, 2020.

[69] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26(1):481–500, 2023.

[70] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018. doi: 10.1109/ITSC.2018.8569938.

[71] Guangxi Lu, Zuobin Xiong, Ruinian Li, Nael Mohammad, Yingshu Li, and Wei Li. Defeat: A decentralized federated learning against gradient attacks. *High-*

Confidence Computing, 3(3):100128, 2023. ISSN 2667-2952. doi: https://doi.org/10.1016/j.hcc.2023.100128. URL https://www.sciencedirect.com/science/article/pii/S2667295223000260.

[72] Massimiliano Luca, Gianni Barlacchi, Bruno Lepri, and Luca Pappalardo. A survey on deep learning for human mobility. *ACM Computing Surveys (CSUR)*, 55(1):1–44, 2021.

[73] Zhenguo Ma, Yang Xu, Hongli Xu, Zeyu Meng, Liusheng Huang, and Yinxing Xue. Adaptive batch size for federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*, 2021.

[74] Daniel Macedo, Danilo Santos, Angelo Perkusich, and Dalton Valadares. A mobility-aware federated learning coordination algorithm. *J. Supercomput.*, 79(17): 19049–19063, may 2023. ISSN 0920-8542. doi: 10.1007/s11227-023-05372-3. URL https://doi.org/10.1007/s11227-023-05372-3.

[75] Daniel Macedo, Danilo Santos, Angelo Perkusich, and Dalton C. G. Valadares. Mobility-aware federated learning considering multiple networks. *Sensors*, 23(14), 2023. ISSN 1424-8220. doi: 10.3390/s23146286. URL https://www.mdpi.com/1424-8220/23/14/6286.

[76] Daniel E. Macedo, Marcus M. Bezerra, Danilo F. S. Santos, and Angelo Perkusich. Orchestrating fog computing resources based on the multi-dimensional multiple knapsacks problem. In Leonard Barolli, editor, *Advanced Information Networking and Applications*, pages 318–329, Cham, 2023. Springer International Publishing. ISBN 978-3-031-28451-9.

[77] Daniel E. Macedo, Danilo F. S. Santos, and Angelo Perkusich. Simulatorfl, 2024. URL https://github.com/enosmacedo/SimulatorFL.

[78] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[79] David Martínez-Rego, Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, and Amparo Alonso-Betanzos. A robust incremental learning method for non-stationary environ-

ments. *Neurocomputing*, 74(11):1800–1808, 2011. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2010.06.037. URL https://www.sciencedirect.com/science/article/pii/S0925231211001007.

[80] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[81] Jed Mills, Jia Hu, and Geyong Min. Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 7(7):5986–5994, 2019.

[82] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017. ISSN 1477-4054. doi: 10.1093/bib/bbx044. URL https://doi.org/10.1093/bib/bbx044.

[83] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*, 6:47980–48009, 2018. doi: 10.1109/ACCESS.2018.2866491.

[84] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne, Jun Li, and H. Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021. doi: 10.1109/COMST.2021.3075439.

[85] Huy T. Nguyen, Nguyen Cong Luong, Jun Zhao, Chau Yuen, and Dusit Niyato. Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6, 2020. doi: 10.1109/WF-IoT48130.2020.9221089.

[86] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[87] Sergey I Nikolenko. *Synthetic data for deep learning*, volume 174. Springer, 2021.

[88] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, pages 1–8, 2018.

[89] Takayuki Nishio and Ryo Yonetani. 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, 2019.

[90] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[91] Chamath Palihawadana, Nirmalie Wiratunga, Anjana Wijekoon, and Harsha Kalutarage. Fedsim: Similarity guided model aggregation for federated learning. *Neurocomputing*, 483:432–445, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.08.141. URL https://www.sciencedirect.com/science/article/pii/S0925231221016039.

[92] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[93] Yuben Qu, Haipeng Dai, Yan Zhuang, Jiafa Chen, Chao Dong, Fan Wu, and Song Guo. Decentralized federated learning for uav networks: Architecture, challenges, and opportunities. *IEEE Network*, 35(6):156–162, 2021. doi: 10.1109/MNET.001.2100253.

[94] Zhe Qu, Xingyu Li, Jie Xu, Bo Tang, Zhuo Lu, and Yao Liu. On the convergence of multi-server federated learning with overlapping area. *IEEE Transactions on Mobile Computing*, 2022.

[95] Prem N. Ramkumar, Heather S. Haeberle, Michael R. Bloomfield, Jonathan L. Schaffer, Atul F. Kamath, Brendan M. Patterson, and Viktor E. Krebs. Artificial intelligence and arthroplasty at a single institution: Real-world applications of machine learning to

big data, value-based care, mobile health, and remote patient monitoring. *The Journal of Arthroplasty*, 34(10):2204–2209, 2019. ISSN 0883-5403. doi: https://doi.org/10.1016/j.arth.2019.06.018. URL https://www.sciencedirect.com/science/article/pii/S0883540319305881.

[96] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021. doi: 10.1109/TNNLS.2020.3015958.

[97] Moamar Sayed-Mouchaweh and Edwin Lughofer. *Learning in Non-Stationary Environments: Methods and Applications*. Springer Publishing Company, Incorporated, 2012. ISBN 1441980199.

[98] Saba Siraj, A Gupta, and Rinku Badgujar. Network simulation tools survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(4):199–206, 2012.

[99] Anand Subramanian. torch_cka, 2021. URL https://github.com/AntixK/PyTorch-Model-Compare.

[100] Shunpu Tang, Wenqi Zhou, Lunyuan Chen, Lijia Lai, Junjuan Xia, and Liseng Fan. Battery-constrained federated edge learning in uav-enabled iot for b5g/6g networks. *Physical Communication*, 47:101381, 2021. ISSN 1874-4907. doi: https://doi.org/10.1016/j.phycom.2021.101381. URL https://www.sciencedirect.com/science/article/pii/S187449072100118X.

[101] Elena Tsiporkova, Michiel De Vis, Sarah Klein, Anna Hristoskova, and Veselka Boeva. Mitigating concept drift in distributed contexts with dynamic repository of federated models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2690–2699, 2023. doi: 10.1109/BigData59044.2023.10386236.

[102] Akhil Vaid, Suraj K Jaladanki, Jie Xu, Shelly Teng, Arvind Kumar, Samuel Lee, Sulaiman Somani, Ishan Paranjpe, Jessica K De Freitas, Tingyi Wanyan, et al. Federated learning of electronic health records to improve mortality prediction in hospitalized

patients with covid-19: Machine learning approach. *JMIR medical informatics*, 9(1): e24207, 2021.

[103] A. Varga. Using the omnet++ discrete event simulation system in education. *IEEE Transactions on Education*, 42(4):11 pp.–, 1999. doi: 10.1109/13.804564.

[104] Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Tarik Taleb. Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. *IEEE Communications Surveys & Tutorials*, 23(2):1342–1397, 2021. doi: 10.1109/COMST.2021.3058573.

[105] Jia Wang, Yazheng Li, Ronghang Ye, and Jianqiang Li. High precision method of federated learning based on cosine similarity and differential privacy. In *2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 533–540, 2022. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00105.

[106] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019. doi: 10.1109/JSAC.2019.2904348.

[107] Dingzhu Wen, Ki-Jun Jeon, and Kaibin Huang. Federated dropout—a simple approach for enabling federated learning on resource constrained devices. *IEEE Wireless Communications Letters*, 11(5):923–927, 2022. doi: 10.1109/LWC.2022.3149783.

[108] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2020.

[109] Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, and Qun Li. A survey of federated learning for edge computing: Research problems and solutions. *High-Confidence Computing*, page 100008, 2021.

[110] Huizi Xiao, Jun Zhao, Qingqi Pei, Jie Feng, Lei Liu, and Weisong Shi. Vehicle selection and resource optimization for federated learning in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2021. doi: 10.1109/TITS.2021.3099597.

[111] Jing Xie, Xiang Yin, Xiyi Zhang, Juan Chen, and Quan Wen. Personalized federated learning with gradient similarity. In *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 268–271, 2021. doi: 10.1109/ICCWAMTIP53232.2021.9674055.

[112] Jinbo Xiong, Renwan Bi, Mingfeng Zhao, Jingda Guo, and Qing Yang. Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles. *IEEE Wireless Communications*, 27(3):24–30, 2020. doi: 10.1109/MWC.001.1900463.

[113] Jinbo Xiong, Renwan Bi, Youliang Tian, Ximeng Liu, and Dapeng Wu. Toward lightweight, privacy-preserving cooperative object classification for connected autonomous vehicles. *IEEE Internet of Things Journal*, 9(4):2787–2801, 2022. doi: 10.1109/JIOT.2021.3093573.

[114] Zuobin Xiong, Wei Li, Qilong Han, and Zhipeng Cai. Privacy-preserving auto-driving: A gan-based approach to protect vehicular camera data. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 668–677, 2019. doi: 10.1109/ICDM.2019.00077.

[115] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. *Journal of Intelligent & Robotic Systems*, 98:297–309, 2020.

[116] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019. ISSN 2157-6904. doi: 10.1145/3298981. URL https://doi.org/10.1145/3298981.

[117] Hao Ye, Le Liang, Geoffrey Ye Li, JoonBeom Kim, Lu Lu, and May Wu. Machine learning for vehicular networks: Recent advances and application examples. *IEEE Vehicular Technology Magazine*, 13(2):94–101, 2018. doi: 10.1109/MVT.2018.2811185.

[118] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. Edgefed: optimized federated learning based on edge computing. *IEEE Access*, 8:209191–209198, 2020.

[119] Rong Yu and Peichun Li. Toward resource-efficient federated learning in mobile edge computing. *IEEE Network*, 35(1):148–155, 2021.

[120] Zhengxin Yu, Jia Hu, Geyong Min, Zhiwei Zhao, Wang Miao, and M Shamim Hossain. Mobility-aware proactive edge caching for connected vehicles using federated learning. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[121] Fotios Zantalis, Grigorios Koulouras, Sotiris Karabetsos, and Dionisis Kandris. A review of machine learning and iot in smart transportation. *Future Internet*, 11(4):94, 2019.

[122] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(3): 2224–2287, 2019. doi: 10.1109/COMST.2019.2904897.

[123] Hangjia Zhang, Zhijun Xie, Roozbeh Zarei, Tao Wu, and Kewei Chen. Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach. *IEEE Access*, 9:98423–98432, 2021. doi: 10.1109/ACCESS.2021.3095915.

[124] Hao Zhang, Qingying Hou, Tingting Wu, Siyao Cheng, and Jie Liu. Data-augmentation-based federated learning. *IEEE Internet of Things Journal*, 10(24): 22530–22541, 2023. doi: 10.1109/JIOT.2023.3303889.

[125] Hongming Zhang and Lajos Hanzo. Federated learning assisted multi-uav networks. *IEEE Transactions on Vehicular Technology*, 69(11):14104–14109, 2020. doi: 10.1109/TVT.2020.3028011.

[126] Hongwei Zhang, Meixia Tao, Yuanming Shi, and Xiaoyan Bi. Federated multi-task learning with non-stationary heterogeneous data. In *ICC 2022 - IEEE International Conference on Communications*, pages 4950–4955, 2022. doi: 10.1109/ICC45855.2022.9838703.

[127] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 9:24462–24474, 2021. doi: 10.1109/ACCESS.2021.3056919.

[128] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.