Budgeted Online Selection of Candidate IoT Clients to Participate in Federated Learning

Ihab Mohammed, *Member, IEEE*, Shadha Tabatabai, *Graduate Student Member, IEEE*, Ala Al-Fuqaha, *Senior Member, IEEE*, Faissal El Bouanani, *Senior Member, IEEE*, Junaid Qadir, *Senior Member, IEEE*, Basheer Qolomany, *Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Machine learning (ML), and deep learning (DL) in particular, play a vital role in providing smart services to the industry. These techniques, however, suffer from privacy and security concerns since data are collected from clients and then stored and processed at a central location. Federated learning (FL), an architecture in which model parameters are exchanged instead of client data, has been proposed as a solution to these concerns. Nevertheless, FL trains a global model by communicating with clients over communication rounds, which introduces more traffic on the network and increases the convergence time to the target accuracy. In this work, we solve the problem of optimizing accuracy in stateful FL with a budgeted number of candidate clients by selecting the best candidate clients in terms of test accuracy to participate in the training process. Next, we propose an online stateful FL heuristic to find the best candidate clients. Additionally, we propose an IoT client alarm application that utilizes the proposed heuristic in training a stateful FL global model based on IoT device-type classification to alert clients about unauthorized IoT devices in their environment. To test the efficiency of the proposed online heuristic, we conduct several experiments using a real data set and compare the results against state-of-the-art algorithms. Our results indicate that the proposed heuristic outperforms the online random algorithm with up to 27% gain in accuracy. Additionally, the performance of the proposed online heuristic is comparable to the performance of the best offline algorithm.**

***Index Terms*—Classification, deep learning (DL), federated learning (FL), Internet of Things (IoT), machine learning (ML), online algorithms, secretary problem.**

Manuscript received July 23, 2020; revised October 2, 2020; accepted October 20, 2020. Date of publication November 5, 2020; date of current version March 24, 2021. *(Corresponding author: Mohsen Guizani.)*

Ihab Mohammed is with the School of Computer Sciences, Western Illinois University, Macomb, IL 61455 USA (e-mail: i-mohammed@wiu.edu).

Shadha Tabatabai is with the Department of Computer Science, Western Michigan University, Kalamazoo, MI 49008 USA, and also with the Computer Science Department, Al-Nahrain University, Baghdad 64074, Iraq (e-mail: shadhamuhinoo.tabatabai@wmich.edu).

Ala Al-Fuqaha is with the Information and Computing Technology Division, Hamad Bin Khalifa University, Doha, Qatar, and also with the Computer Science Department, Western Michigan University, Kalamazoo, MI 49008 USA (e-mail: ala@ieee.org).

Faissal El Bouanani is with the Department of Computer Science, Mohammed V University, Rabat 10000, Morocco (e-mail: f.elbouanani@um5s.net.ma).

Junaid Qadir is with the Department of Electrical Engineering, Information Technology University, Lahore 54000, Pakistan (e-mail: junaid.qadir@itu.edu.pk).

Basheer Qolomany is with the Department of Cyber Systems, College of Business and Technology, University of Nebraska at Kearney, Kearney, NE 68849 USA (e-mail: qolomanyb@unk.edu).

Mohsen Guizani is with the Computer Science and Engineering Department, Qatar University, Doha, Qatar (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2020.3036157

I. Introduction

THE FOURTH industrial revolution (Industry 4.0) promises the provisioning of smart services that enhance the manufacturing process by utilizing emerging technologies, such as Internet of Things (IoT) and artificial intelligence (AI) [1], [2]. In particular, most of the recent advances in Industry 4.0 and AI are driven by machine learning (ML), a branch of AI, and more specifically by deep learning (DL) [1], [3], [4].

The ML and DL techniques, however, require a large amount of data for the training of their models. In particular, serious privacy and security concerns crop up when data are collected and processed from scattered organizations and users [5], [6]. For instance, the prediction of patient mortality using electronic health record (EHR) data dispersed over many hospitals is a complex undertaking due to the various privacy, security, regulatory, and operational issues [7]. Additionally, the communication of potentially large amounts of data from the clients to a central server is costly and can choke the networks when limited bandwidth is available [8]. Such bottlenecks can be observed in vehicular edge computing (VEC) where vehicles have to send their data such as images to roadside servers to build models, which results in the networks being greatly burdened [9].

To address the issues of security, privacy, and excessive communication cost, the technique of federated learning (FL) [10], a distributed ML approach that runs on a server and multiple clients, was proposed. The server and the clients use the same model architecture. The server initiates the global model (i.e., the server model) and executes the following steps over several communication rounds [8] [10].

1) The server sends the global model's parameters to some (or if possible all) clients.
2) Every participating client uses the received global parameters to train the local model using the local data set.
3) Every participating client sends the local model parameters to the server.

2327-4662 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

4) The server aggregates the local parameters received from the clients to update the global model.
5) Eventually, the accuracy of the global model converges to some threshold.

In FL, the server has no access to the client's local data set since only the local model parameters are shared with the server. Consequently, privacy and security are preserved and communication cost is reduced. However, FL suffers from the following two problems [11].

1) Convergence may take a long time, which increases the communication cost.
2) Clients have different computation, storage, and communication resources and different data set sizes, which makes the task of selecting clients a challenge.

FL can be *stateful* or *stateless*. In stateful FL, a candidate client can participate in each of the communication and computation rounds used in training the global model and thus the state is preserved between rounds. Nevertheless, in stateless FL, a candidate client will likely participate in one communication and computation round to train the global model, which means in each round, new fresh candidate clients are utilized [12].

In this article, we propose a stateful FL model with a budgeted number of candidate clients to overcome communication and computation constraints. In other words, from a total of N candidate clients, we select the best $R < N$ candidate clients to participate in training the global model. Now, some candidate clients become available while others become offline or out of communication range over time. Also, we assume that not all candidate clients are available at the same time. Meaning that the problem of selecting R candidate clients is an online problem. As a result, the selection of candidate clients is a challenge. In offline problems, information about all candidate clients are well known in advance rendering the problem of selecting the best R candidate clients trivial. However, in online problems, once a candidate client becomes available then an irrevocable decision must be made on the selection of this candidate client without any prior knowledge about incoming candidate clients. Consequently, we propose a budgeted online selection algorithm that selects the best R candidate clients based on their evaluated test accuracy. The proposed algorithm is inspired by the solution of the secretary problem.

The proposed algorithm can be used in different applications, particularly for online applications with intermittently available mobile clients. Once a client is available, a decision must be made on whether to utilize the client or not since the client may become unreachable like out of communication range or offline [13]. However, once the client is selected, the client will be utilized. Furthermore, decisions cannot be revoked in online applications but might be regretted.

Detection and identification of unauthorized IoT devices are very important especially with the increase in the number of attacks on IoT devices [14]. Therefore, we propose a clients' alarm application that alerts clients about unauthorized IoT devices in their environment. Each client uses a local machine (i.e., server) to monitor the traffic generated by IoT devices in the environment and extract features based on IoT device behavior. Extracted features are used to identify the IoT device type by training an ML model on those features. This is known as *IoT device-type classification*. However, clients cannot identify unknown IoT devices in their environment depending only on the local data set. Therefore, clients subscribe to the alarm service provided by the server on the cloud that utilizes the proposed algorithm. Clients share their model's parameters with the server to train a global model capable of identifying unauthorized IoT devices.

The salient *contributions of this article* are as follows.

1) We propose a model for optimizing accuracy in stateful FL by selecting the best candidate clients based on test accuracy. We formulate the problem of maximizing the probability of selecting the best R candidate clients based on test accuracy from N total candidate clients as a secretary problem and analytically analyze the performance and provide proofs.
2) We propose an online heuristic solution for optimal budgeted client selection based on test accuracy inspired by the secretary problem that works in stateful FL settings. To the best of our knowledge, this is the first work that utilizes online resources selection in FL.
3) We propose a client alarm application for identifying unauthorized IoT devices using the proposed algorithm and IoT device-type classification. We conduct many experiments to evaluate the performance of the proposed heuristic against other state-of-the-art algorithms. Results show an improvement of up to 27% in accuracy compared with the online random algorithm and an accuracy gain of approximately 10% compared with the offline best algorithm.

The organization of the remainder of this article is as follows. A background regarding FL is discussed in Section II. Related literature is reviewed in Section III. Section IV provides a heuristic solution. Section V provides performance proofs including analysis for the worst case scenario. Experimental results are provided in Section VI, where we discuss the application, the data set (and its preprocessing phases), and the conducted experiments. A discussion of the results and the salient lessons learned are provided in Section VII. Finally, this article is concluded in Section VIII by summarizing this work and identifying future research directions.

II. Background

In this section, we introduce the properties and challenges of FL. Then, we review studies related to the online selection of resources using the optimal stopping theory and the secretary problem in particular.

A. Federated Learning

To understand the concepts of FL systems, Li *et al.* [6] provided a comprehensive study of FL systems. They categorize FL systems based on six features, including the ML model, communication architecture, data partition, privacy mechanism, motivation of federation, and scale of federation. Additionally, the authors present a summary of a comparison that includes 42 studies based on the six proposed features.

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

Yang *et al.* [5] and Li *et al.* [6] categorized FL based on data distribution as follows.

1) *Horizontal FL:* Data sets of clients share the same feature space but with a small intersection in regards to the sample space.
2) *Vertical FL:* Data sets of clients share the same sample space but with a small intersection in regards to the feature space.
3) *Hybrid FL (Federated Transfer Learning):* Data sets of clients have a small intersection in regards to both the feature and sample spaces.

We would like to emphasize that a hybrid FL approach is used in this work. The main challenges in implementing FL, as described in [11] and [15], are as follows.

1) *Communication Cost:* There could be many clients (millions) and the system may execute many rounds before converges to the required level of accuracy, which imposes an overload on the network.
2) *Clients Heterogeneity:* The system is heterogeneous and has clients with varying computation, storage, and communication capabilities. Also, the client data sets may differ in features and samples (i.e., the data sets may have statistical heterogeneity).
3) *Privacy and Security:* FL already protects clients' data by only sharing models' parameters. However, sensitive information may be revealed.

Lim *et al.* [15] and Bonawitz *et al.* [16] have highlighted the importance of client selection for enhancing the performance of FL systems since it contributes to both communication cost and resource allocation.

Existing research on enhancing performance in FL follow one of the following approaches.

1) *Algorithm optimization* optimizes the FL algorithm and perform more computation on clients to reduce the convergence time by reducing the number of rounds on the expense of more computation [17]–[24].
2) *Selective updates* select only important updates from the clients or select the best clients in regards to the clients' resources and data size [25]–[30].
3) *Model compression* reduces the amount of data exchanged between clients and the server [18], [31]–[33].

B. Secretary Problem

The secretary problem, which is also known as the marriage problem, dowry problem, beauty contest problem, or Googol is a class of the optimal stopping decision problems. The secretary problem was first introduced by Martin Gardner back in 1960 [34]. The classical secretary problem focuses on the selection of a secretary from a pool of candidates adhering to the following rules [34] [35].

1) The number N of candidates is known.
2) Only one candidate is to be chosen.
3) Candidates are interviewed sequentially in a random order.
4) Each candidate must be accepted or rejected before interviewing the next one (with no provision for recalling rejected candidates later).
5) Candidates are ranked from best to worst and the decision of accepting or rejecting a candidate depends on the relative ranks of candidates interviewed so far.
6) The problem focuses on maximizing the probability of selecting the best candidate.

The solution of the secretary problem is for some integer $1 \leq \alpha < N$, reject the first α candidates then select the first candidate with rank better than of those observed candidates. The goal is to find the optimal α that maximizes the probability of selecting the best candidate. Actually, it has been proven that the optimal value for α is 0.367879 with optimal probability of $(1/e)$ [34]. In other words, the probability of finding the best candidate is 37% when rejecting the first 37% of candidates and selecting the first candidate with ranking better than those observed ones.

Freeman [36] reviewed the extensions and generalizations of the secretary problem. They indicate that some researchers focus on the secretary problem when the number of candidates is unknown. Other researchers assume that candidates' ranks follow a specific distribution such as Poisson. Additionally, they show that some studies focus on selecting R candidate instead of one.

In this article, we are interested in studies of the secretary problem where R top candidates are selected. Gilbert and Mosteller [37] provided many variations of the secretary problem studied under different assumptions and one of these cases is for selecting R candidates with one of the candidates as the best candidate. Kleinberg [38] proposed an algorithm to maximize the sum of ranks of the R selected candidates. The algorithm has two stages. In the first stage, the classical secretary algorithm is recursively applied on roughly the first half of candidates to select $l = R/2$ best candidates. In the second stage, the rank of the lth selected candidate in the first stage is used as a threshold for selecting $R/2$ candidates from the second half of candidates. The author states that the algorithm has a competitive ratio of $1 - O(\sqrt{q/R})$. Babaioff *et al.* [39] proposed an algorithm to maximize the sum of the R selected candidate. The algorithm rejects the first $\lfloor n/e \rfloor$ candidates and records the R highest rankings in set S. Next, when a candidate with a rank higher than the minimum rank in S is encountered, the candidate is selected and the minimum rank in S is removed. This is repeated until either S is empty or all candidates are reviewed. The authors indicate that the algorithm has a competitive ratio no worse than e for all values of R.

The work in this article is inspired by the aforementioned studies. However, this work is different in that we find the optimal stopping position α, which we call α^*, to maximize the probability of selecting the R top candidates. We reject the first α^* candidates and record the best rank. Then, we use the best rank as a threshold in selecting the top R candidates.

III. Related Work

FL is a hot research area that has recently grabbed the attention of many researchers. In this section, we list different

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

approaches for enhancing the performance and discuss studies in each approach.

A. Algorithm Optimization

Some researchers work on optimizing the algorithm used in FL to reduce the convergence time and thus reduce the generated traffic in the network. Replacing the minibatch stochastic gradient descent (mb-SGD) optimization model with *Adam* has been studied in [18]. The authors propose CE-FedAvg, an algorithm that uses Adam optimization and compresses models before uploading to the server. The authors claim that using Adam optimization along with model compression reduces the convergence time by reducing the number of rounds and the amount of data exchanged between clients and the server. Using a multiobjective evolutionary algorithm with neural networks in FL has been studied in [19]. The authors use the elitist nondominated sorting genetic algorithm (NSGA-II) to minimize the communication cost at the expense of higher computation cost. Liu *et al.* [20] proposed momentum FL (MFL), which uses momentum gradient descent (MGD) in every step of local updates rather than the first-order gradient descent. The authors state that since MGD considers preceding iteration, it converges faster than the traditional FL system.

Other researchers proposed algorithms that utilize the computation power on clients' machines to speed up the convergence process. To reduce the number of rounds, Liu *et al.* [21] proposed to use federated stochastic block coordinate descent (FedBCD) algorithm in vertical FL, which let clients do multiple local model updates before syncing with each other. Yao *et al.* [22] claimed that using two models in every client instead of a single model can reduce the number of rounds. Besides training the global model received from the server, each client trains another local model and uses the maximum mean discrepancy (MMD) between the output of the two models. Using agents on edge nodes between clients and the server are studied in [23] and [24]. Multiple agents perform partial model aggregation before communicating with the server to reduce the communication cost between clients and the server.

Other researchers study the tradeoff between the number of iterations performed by clients to minimize the loss function and the frequency of global aggregation done by the server. Wang *et al.* [17] computed the convergence bound of the gradient-descent algorithm then designed an algorithm that finds the best frequency of global aggregation based on system dynamics, model characteristics, and data distribution to minimize the consumed computation and communications. Qolomany *et al.* [40] proposed a particle swarm optimization (PSO)-based technique to optimize the hyperparameter settings for the local ML models in an FL environment. They evaluated and compared the proposed PSO-based parameter optimization approach with the grid search technique. They found that the number of client–server communication rounds to explore the landscape of configurations to find the near-optimal parameter settings is greatly decreased by two orders of magnitude using the PSO-based approach compared to the grid search method. To deal with heterogeneous data inherent in federated networks, Li *et al.* [41] proposed a modified version of FedAvg; namely, FedProx, that allows for variable amounts of work to be performed locally across devices, and relies on a proximal term which helps to improve the stability of the method against heterogeneous data.

B. Selective Updates

Chen *et al.* [25] formulated a client selection and resource allocation optimization problem for FL in wireless networks to minimize the value of the loss function. They first derived an equation to represent the expected convergence rate of the FL algorithm. Next, they simplified the optimization problem as a mixed-integer nonlinear programming problem. Then for a given uplink resource block allocation and client selection, they compute the optimal transmit power. Finally, they transform the problem into a bipartite matching problem and use the Hungarian algorithm to find the optimal client selection and resource block allocation. Nishio and Yonetani [26] proposed a new FL protocol named FedCS to enhance the efficiency of FL. The basic idea of the proposed protocol is to select clients based on their computation/communication capabilities and their data size instead of picking clients randomly. To reduce the communication overload, Wang *et al.* [27] proposed an approach that identifies clients with irrelevant updates and prevent those clients from uploading their updates to the server. Anh *et al.* [28] and Nguyen *et al.* [29] proposed the selection of clients based on the consumed energy in model's transmission and training, clients' distance from the server, and channel availability using the deep reinforcement learning (DRL) approach. Yoshida *et al.* [30] proposed a hybrid FL approach based on the assumption that some clients share and upload their data to the server to improve the accuracy and mitigate the degradation resulted from nonindependent and identically distributed (nonIID) data. However, uploading clients' data to the server violates the rules of FL.

C. Model Compression

Sattler *et al.* [31] proposed a new compression framework named sparse ternary compression (STC). The authors claim that their compression framework performs better than other proposed methods in the literature in bandwidth-constrained learning environment. Konecný *et al.* [32] proposed to use structured updates (low rank and random mask) and force models to use these structures and also sketched updates with lossy compression before sending models to the server. On the other hand, Caldas *et al.* [33] applied lossy compression on the model sent from the server to the clients.

The related research discussed above has a high computational cost. The clients' intensive computation and algorithm optimization approach requires intensive computation. In addition to the extra computation required by the compression approach, it is best applied to models with large parameter vector such as images or models with many hidden layers. The presented studies using the selective updates approach either are too difficult to train (especially when a large number of clients are used as in DRL) or have no analysis and/or proofs for convergence. In contrast, our proposed algorithm, which uses a selective updates approach, does not require intensive

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF MATHEMATICAL NOTATIONS

Notation	*Definition*
N	Total number of candidate clients arriving until time T. Each candidate client is identified by an index in the interval $1..N$
K	Number of communication rounds
E	Number of epochs
R	Number of required best candidate clients
$(\mathcal{C}_\ell)_{1\leq\ell\leq N}$	Set of candidate clients
α	An index in the interval $1..N$
α^*	The optimal value of α
$\mathcal{C}_M$	The best candidate client in $[1..\alpha]$, i.e., $1 \leq M \leq \alpha$
$(i_m)_{1\leq m\leq R}$	The set of R positions corresponding to the top R best candidate clients in the interval $[\alpha+1..N]$, better than $\mathcal{C}_M$, such that $\alpha+1 \leq i_m \leq N$ and $\mathcal{C}_{i_m}$ is worst than $\mathcal{C}_{i_{m-1}}$ for all $2 \leq m \leq R$
$\{\mathcal{A}^{(i)}\}_{\alpha+1\leq i\leq N}$	The set of events where "the ith candidate better than $\mathcal{C}_M$ is selected"
$\{\mathcal{B}^{(m,i)}\}_{1\leq m;\alpha+1\leq i\leq N}$	The set of events where "the ith candidate is the mth best one"
$\mathcal{E}_R$	The event "The R best candidates in $[\alpha+1..N]$ better than $\mathcal{C}_M$ are selected" occurring with the probability $\Pr(\mathcal{E}_R)$; Obviously, $\bigcap_{\ell=1}^{R} \mathcal{E}_R = \bigcup_{i_R\leq i_{R-1}..\leq i_1} \bigcap_{\ell=1}^{R} \left\{\mathcal{A}^{(i_\ell)} \cap \mathcal{B}_R^{(\ell,i_\ell)}\right\}$ with i_1 and i_R correspond to the best and the worst combinations, among R selected ones, respectively
r_1	Minimum number of best candidate clients.
r_2	Maximum number of best candidate clients.
$\mathcal{P}^{(r_1,r_2)}$	Probability to select the R best candidates $(i_m)_{1\leq m\leq R}$ where $r_1 \leq R \leq r_2$ and $r_1, r_2 < N$. $\mathcal{P}^{(r_1,r_2)} = \sum_{R=r_1}^{r_2} \Pr(\mathcal{E}_R)$.
P_i	Probability of selecting candidate i

computations or a large parameter vector, and we also provide analysis and proofs demonstrating convergence.

IV. PROPOSED CLIENT SELECTION SOLUTION

A. System Model

We assume *N* candidate clients and one server. Also, we assume a budget of *R* candidate clients. The nature of the proposed model is online since some clients become available while others become unreachable or offline over time. Consequently, the server must make an irrevocable decision to accept (i.e., select) or reject a candidate client once a candidate client becomes available. The server runs the proposed heuristic (explained in the next section), which initializes the global model, selects the *R* best candidate clients based on their test accuracy, and then trains the global model using the selected candidate clients in *K* communication rounds. Each selected candidate client trains the local model in *E* epochs using the local data set but with the global model parameters. Moreover, we assume the data sets of candidate clients are different in size. Therefore, we use the terms *fat clients* and *thin clients* to point to candidate clients with different sizes of data sets. We note that in some literature, the terminology of elephants (instead of fat) and mice (instead of thin) is used instead [42]. For the convenience, we have listed the main mathematical notations used in this article in Table I.

B. Problem Formulation

The problem we tackle in this article is to select the best set of candidate clients that provide higher test accuracy when training the global model using their local data set. This problem is similar to the famous *secretary problem*, which aims to maximize the probability of selecting the maximum element from a randomly ordered sequence [39]. The secretary problem is formulated as a linear programming problem as follows [43]:

$$\begin{aligned} \max \quad & \frac{1}{N} \cdot \sum_{1=1}^{N} iP_i \\ \text{s.t.} \quad & \forall 1 \leq i \leq N \quad i \cdot P_i \leq 1 - \sum_{j=1}^{i-1} P_j \\ & \forall 1 \leq i \leq N \quad P_i \geq 0. \end{aligned}$$

In the traditional secretary problem, the objective function aims to maximize the probability of selecting the best candidate. However, instead of selecting one element, in this problem, *R* elements must be selected. The secretary problem is one scenario of the *optimal stopping theory*. In the secretary problem, an employer wants to hire a secretary and there are *N* candidates. The employer cannot assess the quality of a candidate until after the end of the interview and have to make an irrevocable hiring decision. Thus, the employer may end up hiring a candidate before interviewing the rest of the candidates and the hiring of the best candidate is not guaranteed.

Our solution is inspired by the *secretary problem*. The quality of a client is determined by its test accuracy. We evaluate the test accuracy (i.e., quality) of the first α^* (see Section V) clients and reject them all. Then, select the next *R* clients with test accuracy better than the best test accuracy of the first α^* clients, and if none is found then select the last clients.

C. Proposed Algorithm

The proposed heuristic identifies the best test accuracy among the first few available candidate clients then use this test accuracy as a threshold for accepting or rejecting candidate clients available later. The heuristic accepts the parameters *N*, *R*, r_1, r_2, *K*, *E*, and δ as explained in Algorithm 1 and consists of *three stages* that run every δ time units to update the global model.

In the *first stage* (Algorithm 1, lines 2–11), the value of α^* (discussed in Section V) is computed based on the value of r_1 and r_2 using (7). The first α^* candidate clients that are available are then tested to determine the best test accuracy. However, none of those candidate clients are accepted. Whenever a candidate client becomes available, the server initializes the global model and communicates with the candidate client to evaluate its test accuracy. Testing is performed by sending the initialized global model's parameters from the server to the candidate client for one communication round so that the candidate client trains the local model with these parameters using the local data set. Then, the candidate client sends back the updated parameters to the server. The server

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Proposed Heuristic

Input: N (expected number of clients), R (number of selected candidate clients), r_1, r_2 (to compute α^*), K (number of communication rounds), E (number of epochs per client), and δ
Output: Trained global model
1: **for** every δ time units **do**
// Find best test accuracy for first m candidate clients, m:1..α^*
2: Initialize the global model
3: Compute α^* based on r_1, r_2, and N using equation (7)
4: Set A_b, best test accuracy = 0
5: **for** m = 1 to α^* **do**
6: Test client $\mathcal{C}_{\mathcal{M}}$ and record A_m, its test accuracy
7: **if** $A_m > A_b$ **then**
8: Set $A_b = A_m$
9: **end if**
10: Reject candidate client C_m
11: **end for**
// Find R best candidate clients
12: Set S_b, set of best candidate clients = []
13: Set N_b, number of best candidate clients found = 0
14: **for** $m = \alpha^* + 1$ to N **do**
15: **if** $N_b = R$ **then**
16: Reject candidate client C_m
17: **else if** $(N - m) \leq (R - N_b)$ **then**
18: Accept candidate client C_m and add it to S_b
19: Increment N_b by 1
20: **else**
21: Test client C_m and record A_m, its test accuracy
22: **if** $A_m > A_b$ **then**
23: Accept candidate client C_m and add it to S_b
24: Increment N_b by 1
25: **end if**
26: **end if**
27: **end for**
// Start training
28: **for** $k = 1$ to K **do**
29: Send global model to all candidate clients in S_b
30: Candidate clients train global model on local dataset for E epochs
31: Server average aggregated model parameters from candidate clients in S_b
32: **end for**
33: **end for**

evaluates the received parameters (i.e., no averaging is applied since only one candidate client is involved) using the test data set to determine the test accuracy of the candidate client. After testing α^* candidate clients, the server selects the best test accuracy to be used as a threshold in the second section.

In the *second stage* (Algorithm 1, lines 12–27), whenever a candidate client becomes available, it gets tested in the same way explained in the first section. Next, the server accepts (i.e., selects) the candidate client only if its test accuracy is greater than the best test accuracy found in the first section. Nonetheless, if the number of available candidate clients is less than the number of required candidate clients (i.e., R) then the server has no choice but to select those remaining candidate clients. In the worst case scenario, the candidate client with the best test accuracy is met during the first section. Consequently, all candidate clients met early in the second stage are rejected for having a test accuracy less than that of the best test accuracy found in the first section. As a result, the server is forced to accept all candidate clients that are met at the end of the second section. In fact, in the worst case scenario, the proposed heuristic behaves similar to the random algorithm explained in Section VII.

In the *third stage* (Algorithm 1, lines 28–32), once the best candidate clients are identified, the global model is trained using the selected candidate clients for K communication rounds as described in Section I.

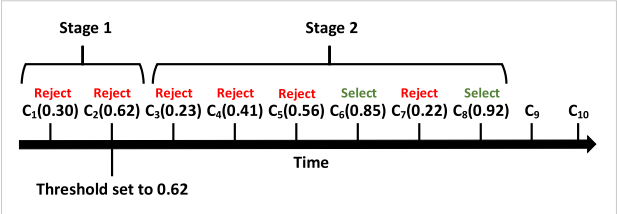

Fig. 1. Illustrative example of the proposed algorithm.

D. Illustrative Example

To understand the proposed algorithm in more depth, we present an example where we observe one run cycle (when δ is 1) of the proposed algorithm overtime (see Fig. 1). Assume a total of ten candidate clients becoming available over time during the observed period (i.e., $N = 10$). We refer to candidate i as $\mathcal{C}_i$. Additionally, we set the budget to 2 candidate clients (i.e., $R = 2$), which means that we want to select the best two candidate clients for training the global model. Moreover, we set E, the number of epochs, to 3 and set K, the number of communication rounds between the server, and selected candidate clients for training the global model to 20. Also, the value of α^* is computed based on (7) in Section V-A (assuming r_1 is 1 and r_2 is 2) and its value is 2.

When a candidate client becomes available then 1) the proposed algorithm initiates the global model's parameters then sends them to the candidate client; 2) the candidate client trains the local model for E epochs using the received parameters from the server on the local data set; 3) the candidate client sends the updated parameters to the server; and 4) the server evaluates the accuracy of the candidate client by testing the global model using the updated parameters on the test data set. Then, the proposed algorithm must make an irrevocable decision on whether to use this client or not based on its evaluated test accuracy.

The proposed algorithm runs in three stages. In the first stage, the proposed algorithm communicates with the first α^* (i.e., 2) candidate clients and evaluate their test accuracy to determine the best test accuracy, which is used as a selection threshold with the rest of candidate clients that become available later. Thus, when $\mathcal{C}_1$ becomes available, the proposed algorithm communicates with $\mathcal{C}_1$ then evaluates its test accuracy and finds it 0.30. The proposed algorithm sets its selection threshold to 0.30 and rejects $\mathcal{C}_1$. Next, $\mathcal{C}_2$ becomes available and the proposed algorithm communicates with $\mathcal{C}_2$ then evaluates its test accuracy and finds it 0.62. The proposed algorithm updates its selection threshold to 0.62 as illustrated in Fig. 1 where $\mathcal{C}_i(x)$ represents candidate client i with evaluated test accuracy x (test accuracy is a number between 0 and 1, where 0 means the trained model fails to identify all test samples while 1 means the trained model identifies all test samples successfully).

In the second stage, the proposed algorithm will continue to communicate with any candidate client that becomes available and evaluate its test accuracy to decide on the selection of this

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

candidate client. This process continues as shown in Fig. 1 until the proposed algorithm selects two candidate clients and as follows.

1) $\mathcal{C}_3$ becomes available and its test accuracy is 0.23 and thus gets rejected.
2) $\mathcal{C}_4$ becomes available and its test accuracy is 0.41 and thus gets rejected.
3) $\mathcal{C}_5$ becomes available and its test accuracy is 0.56 and thus gets rejected.
4) $\mathcal{C}_6$ becomes available and its test accuracy is 0.85 and thus gets selected.
5) $\mathcal{C}_7$ becomes available and its test accuracy is 0.2 and thus gets rejected.
6) $\mathcal{C}_8$ becomes available and its test accuracy is 0.92 and thus gets selected.
7) The server is not going to communicate with $\mathcal{C}_9$ and $\mathcal{C}_{10}$ once they are available since the proposed algorithm has already selected two candidate clients.

In the third stage, the proposed algorithm trains the global model using $\mathcal{C}_6$ and $\mathcal{C}_8$ with K communication rounds but without initiating the global model in every round. A best case scenario is presented in this example, but a worst case scenario can occur if the test accuracy of $\mathcal{C}_2$ is evaluated and found as 0.93. In this case, the proposed algorithm rejects both $\mathcal{C}_6$ and $\mathcal{C}_8$. Eventually, the proposed algorithm will have to communicate with the last two clients ($\mathcal{C}_7$ and $\mathcal{C}_{10}$) and selects both.

V. Performance Analysis

The performance of the proposed algorithm explained in the previous section depends vitally on the optimal value of α, which is α^*. In this section, we derive an equation for computing the value of α^* and prove its validity. This equation is plugged in the first stage of the proposed algorithm as mentioned in Section IV-C. Finally, we analytically analyze the performance of the proposed algorithm in worst case scenario.

A. Optimal Value for α

By assuming 1) M and i_m positions are not known in advance; 2) the candidates can arrive in any order; and 3) $N, \alpha >> R$, we aim to find the optimum value α^*, depending on both, allowing to maximize $\mathcal{P}^{(r_1,r_2)}$.

Lemma 1: The following summation:

$$\mathcal{K}(R,\alpha) = \sum_{i_R=\alpha+1}^{N-R+1} \frac{1}{i_R-1} \sum_{i_{R-1}=i_R+1}^{N-R+2} \frac{1}{i_{R-1}-1} \cdots \sum_{i_1=i_2+1}^{N} \times \frac{1}{i_1-1} \tag{1}$$

can be tightly approximated by

$$\mathcal{K}(R,\alpha) \approx \frac{\left(\log \frac{N}{\alpha}\right)^R}{R!}. \tag{2}$$

Proof: Let us proceed by induction. one can ascertain that for $R = 1$, the summation $\sum_{i_1=\alpha+1}^{N} [1/(i_1-1)]$ can be approximated by the $\int_\alpha^N (dt/t) = \log(N/\alpha)$, confirming (2).

Let us assume that (2) holds for $R-1$. One obtains

$$\begin{aligned}\mathcal{K}(R,\alpha) &= \sum_{i_R=\alpha+1}^{N-R+1} \frac{1}{i_R-1}\mathcal{K}(R-1,i_R) \\ &\approx \frac{1}{(R-1)!}\sum_{i_R=\alpha+1}^{N-R+1} \frac{\left(\log \frac{N}{i_R}\right)^{R-1}}{i_R-1} \\ &\approx \frac{1}{(R-1)!}\int_\alpha^{N-R+1} \frac{\left(\log \frac{N}{t}\right)^{R-1}}{t} dt. \end{aligned} \tag{3}$$

Finally, taking into account that $R << N$ (i.e., $N-R+1 \approx N$), it follows that:

$$\mathcal{K}(R,\alpha) \approx \frac{1}{R!}\left[-\left(\log \frac{N}{t}\right)^R\right]_\alpha^N \tag{4}$$

$$\approx \frac{1}{R!}\left(\log \frac{N}{\alpha}\right)^R \tag{5}$$

which concludes the proof. ∎

Proposition 1: For all positive numbers $r_1, r_2 << \alpha, N$, the approximation

$$\mathcal{P}^{(r_1,r_2)} \approx \frac{\alpha}{N}\sum_{R=r_1}^{r_2} \frac{1}{R!}\left(\log \frac{N}{\alpha}\right)^R \tag{6}$$

holds, and the optimum value maximizing such probability is

$$\alpha^* = N\exp\left(-\left(\frac{r_2!}{(r_1-1)!}\right)^{\frac{1}{r_2-r_1+1}}\right). \tag{7}$$

Proof: Given that the indices of the selected candidates are sorted in increasing order of candidates' accuracies, $\mathcal{E}_R$ can be broken into R exclusive events as follows.

1) Candidate client M is the best one in $[1, i_R-1]$.
2) Candidate clients i_m are the best ones in $[1, i_{m-1}-1]$, $2 \le m \le R$.
3) Candidate client i_1 is the best one in $[1, N]$.

Consequently

$$\Pr(\mathcal{E}_R) = \sum_{i_R=\alpha+1}^{N-R+1} \sum_{i_{R-1}=i_R+1}^{N-R+2} \times \cdots \sum_{i_1=i_2+1}^{N} \Pr\left(\underbrace{\bigcap_{\ell=1}^{R}\left\{\mathcal{A}^{(i_\ell)} \cap \mathcal{B}_R^{(\ell,i_\ell)}\right\}}_{\mathcal{D}_R}\right). \tag{8}$$

With the aid of Bayes's rule, $\mathcal{D}_R$ can be rewritten as

$$\Pr(\mathcal{D}_R) = \Pr\left(\mathcal{A}^{(i_R)}\middle|\mathcal{B}_R^{(R,i_R)} \cap \mathcal{D}_{R-1}\right)\Pr\left(\mathcal{B}_R^{(R,i_R)}\middle|\mathcal{D}_{R-1}\right). \tag{9}$$

The probability to select the Rth best one among $[1 \cdots N] \setminus \{i_1, i_2, \ldots, i_{R-1}\}$ is

$$\Pr\left(\mathcal{B}_R^{(R,i_R)}\middle|\mathcal{D}_{R-1}\right) = \frac{1}{N-R+1} \tag{10}$$

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CHOOSING α^* THAT MAXIMIZES THE PROBABILITY TO SELECT R BEST CANDIDATES SUCH THAT $r_1 \leq R \leq r_2$ AND $N = 1000$

r_1	r_2	α^*	Percentage (%) $x^* = \frac{\alpha^*}{N}$	$\mathcal{P}_{\max}^{(r_1,r_2)}$
2	2	135.3353	13.53	0.2707
2	3	49.7871	4.97	0.4481
2	4	0.3355	0.03	0.0966
3	3	49.7871	4.97	0.2240
3	4	2.4788	0.24	0.2231

with the conditional probability in (9) can be evaluated as

$$\Pr\left(\mathcal{A}^{(i_R)} \middle| \mathcal{B}_R^{(R,i_R)} \cap \mathcal{D}_{R-1}\right) = \frac{\alpha}{i_R - 1} \prod_{\ell=1}^{R-1} \frac{1}{i_\ell - 1}. \quad (11)$$

Substituting (11), (10), and (9) into (8), one obtains

$$\Pr(\mathcal{E}_R) = \frac{\alpha}{N - R + 1} \mathcal{K}(R, \alpha). \quad (12)$$

Leveraging Lemma 1 and noting that $N - R + 1 \approx N$, (6) is obtained. Now, defining $x = \alpha/N$ (i.e., $0 \leq x \leq 1$), the two first derivatives of $\mathcal{P}^{(r_1,r_2)}$ with respect to x can be expressed as

$$\frac{\partial \mathcal{P}^{(r_1,r_2)}}{\partial x} = \frac{(-\log x)^{r_2}}{r_2!} - \frac{(-\log x)^{r_1-1}}{(r_1-1)!} \quad (13)$$

$$\frac{\partial^2 P^{(r_1,r_2)}}{\partial x^2} = -\frac{1}{x}\left[\frac{(-\log x)^{r_2-1}}{(r_2-1)!} - \frac{(-\log x)^{r_1-2}}{(r_1-2)!}\right]. \quad (14)$$

Thus, by solving $[(\partial \mathcal{P}^{(r_1,r_2)})/\partial x] = 0$ and setting $\alpha^* = Nx^*$, we get (7). Moreover, it can be easily checked that the second derivative evaluated at x^*

$$\frac{\partial^2 P^{(r_1,r_2)}}{\partial x^2} = -\frac{(-\log x)^{r_1-2}}{x^*(r_1-2)!}\left[\frac{(r_1-2)!}{(r_2-1)!}\underbrace{\left(-\log x^*\right)^{r_2-r_1+1}}_{=\frac{r_2!}{(r_1-1)!}} - 1\right]$$

$$= -(r_2 - r_1 + 1)\frac{(-\log x)^{r_1-1}}{x^*(r_1-1)!} \quad (15)$$

is negative as $r_2 > r_1$ and $x^* \leq 1$, which completes the proof. ■

Table II summarizes some values of the optimal number α^* along with the aforementioned maximum probability for various values of r_1 and r_2, when $N = 1000$. Note that the probability (6) is an increasing function on r_2, while its maximum value is not monotone as it depends also on r_1 as summarized in Table I. It can be seen also that:

1) the smaller r_1 is, the greater the optimal value ($\alpha^* = Nx^*$);
2) for a fixed r_1, the larger r_2 is, the smaller α^*.

Fig. 2 shows that the probability of selecting the best R clients is higher when the value of α is small.

B. Worst-Case Analysis (Competitive Ratio Analysis)

The worst case scenario is encountered when the proposed heuristic does not find candidates that exceed $\mathcal{C}_{\mathcal{M}}$ from index α^* until N. The competitive ratio in the worst case scenario is computed over all possible input sequences as the maximum ratio of the gain of the online algorithm and the optimal offline algorithm [44].

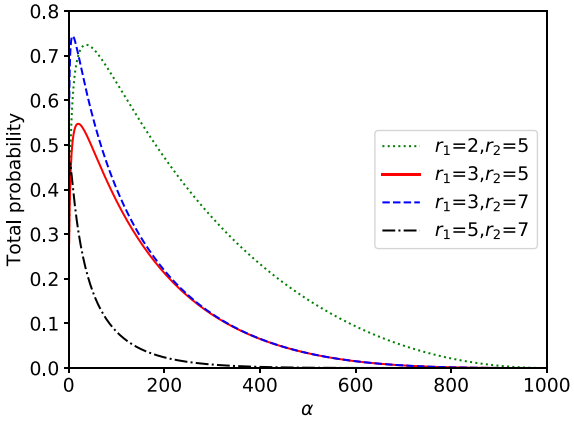

Fig. 2. Effects of α on the probability of selecting the best clients.

Proposition 2: The heuristic's worst case performance has a competitive ratio of $O(1)$ when R is proportional to N.

Proof: Let ALG be the proposed heuristic and OPT be the optimal algorithm.

The worst case happens when the highest element appears before index α^*. In that case, the proposed algorithm randomly selects candidate clients from index α^*+1 until N. A candidate client within this range of indices is selected with probability $[R/(N - \alpha^*)]$. Consequently, the following proof is concluded as follows:

$$\text{Com}_r = \frac{ALG}{OPT} = \frac{R}{N - \alpha^*}.$$

Thus, Com_r, the competitive ratio, becomes $\mathcal{O}(1)$ when R is proportional to N. ■

VI. EXPERIMENTAL SETTINGS

In this section, we describe the application proposed in this article in detail first. Next, we describe the data set used in the simulation and describe the data set preparation phases used to transform the raw data set into N candidate clients' data sets. Finally, we discuss conduced experiments.

A. Use Case: IoT Device-Type Classification

IoT devices perform specific tasks, which makes their network behavior predictable [45]. There are plenty of studies on IoT device-type classification or fingerprinting in [14] and [45]–[53]. Those studies concentrate on identifying IoT devices type for different reasons, including security, access control, provisioning, resource allocation, and management [46]. Actually, most of those studies concentrate on security in response to recent incidents [54], [55]. In one incident, thousands of IoT devices including surveillance cameras are used for Distributed Denial of Service (DDoS) attack [14]. Therefore, we propose a client alarm application based on IoT device-type classification in FL settings to identify unauthorized IoT devices. The IoT device-type classification is

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

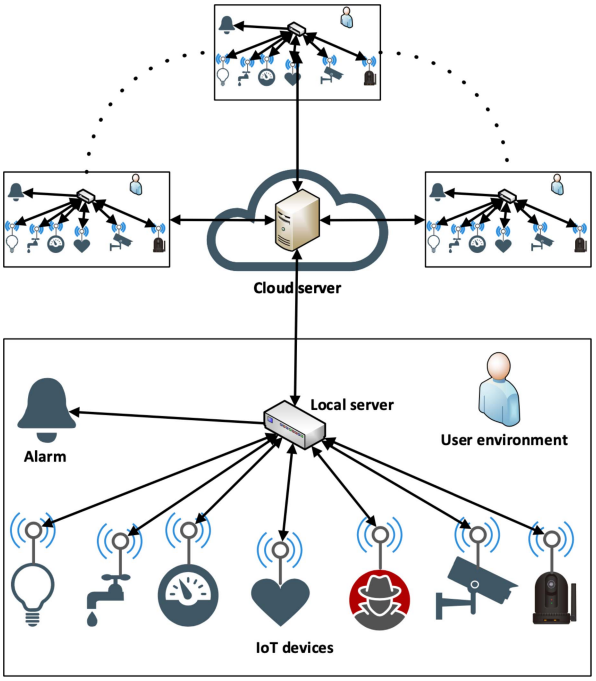

Fig. 3. Illustration of the clients alarm application. The cloud server running the proposed algorithm communicates with the local servers of the best subscribed clients to train the global model.

inspired by the work in [53]. We aim to use the proposed application as a use case to test the performance of the proposed heuristic.

The proposed application consists of N candidate clients, a main server in the cloud, and an alarm mechanism. Each client's environment has several IoT devices, a local machine (i.e., the local server), and an alarm device as shown in Fig. 3. The alarm can be a physical device or software that delivers e-mail, text messages, or any other form of notification to the client. The local server monitors the traffic generated by IoT devices, extracts features, and builds a local data set. Then, train the local model using the local data set. However, training on local data set is not sufficient to identify unknown IoT devices in the environment. As a result, the clients subscribe to the alarm service provided by the server through the use of FL. The server is responsible for running the proposed FL algorithm. Also, the server and clients cooperate to build a global model capable of classifying devices used by participating clients. In other words, clients can use the global model to identify unknown devices from the knowledge of other clients.

The proposed algorithm in the server trains a global model by sharing only the model's parameters with clients and thus preserving the security and privacy of clients. The process of training the global model is repeated every δ time units to make sure that the new clients, and clients with the new installed IoT devices, are considered and included.

Employing all clients in the training process produce high traffic, which overloads the network. Additionally, this might be infeasible since some clients are not available all the time. However, selecting clients with high accuracy contribution to the global model training enhance the classification accuracy, which is done by the proposed heuristic.

B. Data Set Details and Preprocessing Phases

To test the performance of the proposed heuristic, we use a real data set collected by researchers from the University of New South Wales (UNSW), Sydney, Australia [53]. The data set is created using 28 IoT devices and also some nonIoT devices installed in a lab on the campus of the university. Trace data are captured over six months between October 1, 2016 and April 13, 2017. However, only 20 days of trace data are available for the public. Raw data consisting of packet headers and payload information are captured using the `tcpdump` tool installed on the gateway. The data set is available as a set of pcap (packet capture) files and also as a set of CSV (comma-separated values) files. The data set consists of 20 pcap files, one file per day.

The raw data set is processed in five phases (illustrated in Fig. 4) in order to create N candidate client's data sets to simulate FL settings as described next.

In the *flows collection phase* (Phase-1), we collect flows from raw data in pcap files using the `joy` tool developed by Cisco Systems [56]. `Joy` is a data collection tool that reads the data from raw traffic (or from pcap files) and produces a JavaScript object notation (JSON) file with a summary of the traffic data in the form of flows. We create a bash script that uses the `joy` tool to process the pcap files and produce JSON files. Each JSON file contains flows related to a specific IoT device based on the MAC address listed in Table III, which includes names of devices and their MAC addresses as indicated in the data set's website [53]. To filter by MAC address, we use the Berkeley/BSD Packet Filter syntax supported by the `joy` tool through the data feature options. Each flow in the resultant JSON file has a flow key that includes the source and destination addresses, and the source and destination port and protocol numbers. Each flow also contains number of bytes, number of packets, start time, and end time. Additionally, `joy` can be configured to save more information per flow. Algorithm 2 describes the flow collection process. Also, the script is available on GitHub [57].

In the *features extraction phase* (Phase-2), we extract features from the flows stored in JSON files. Inspired by a previous study [53], we analyze the flows and extract features as listed in Table IV. Features are saved in a CSV file with the first ten columns for features and the last column for the labels, which are the IoT device IDs. Algorithm 3 shows the steps used in the extraction process. In addition, the Python code for extracting the features is made available on GitHub [57].

In the *normalization phase* (Phase-3), we first normalize all features by transforming features' values to be between 0 and 1 using the `MinMaxScaler` function from the `scikit-learn` library [58]. Second, to ensure that samples are distributed randomly, we randomly reindex all normalized features in the data set.

In the *ML model design phase* (Phase-4), we split the data set into two parts: 1) the training data set (80% of the original) and 2) the test data set (20% of the original). To ensure a fair comparison between the proposed algorithm and other algorithms, the test data set is stored on the server. We then design a deep neural network (DNN)-based ML model with

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

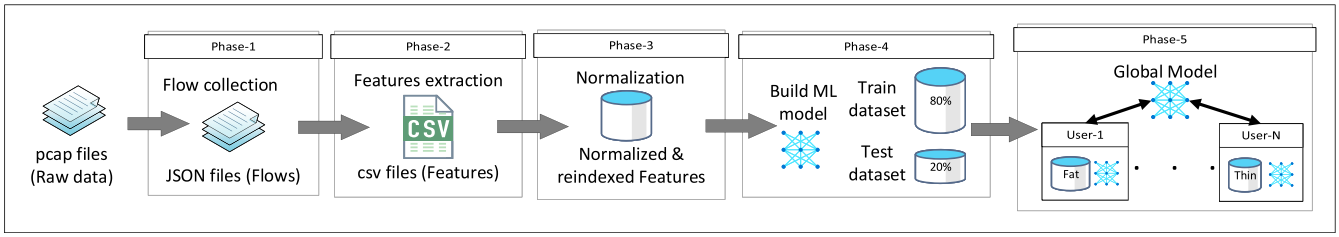

Fig. 4. Data set preprocessing phases (through which the raw data set is transformed to the N candidate clients' data sets).

TABLE III
NAMES AND MAC ADDRESSES OF THE USED IOT DEVICES

IoT device name	*MAC address*
Amazon Echo	44:65:0d:56:cc:d3
August Doorbell Cam	e0:76:d0:3f:00:ae
Awair air quality monitor	70:88:6b:10:0f:c6
Belkin Camera	b4:75:0e:ec:e5:a9
Belkin Motion Sensor	ec:1a:59:83:28:11
Belkin Switch	ec:1a:59:79:f4:89
Blipcare BP Meter	74:6a:89:00:2e:25
Canary Camera	7c:70:bc:5d:5e:dc
Dropcam	30:8c:fb:2f:e4:b2
Google Chromecast	6c:ad:f8:5e:e4:61
Hello Barbie	28:c2:dd:ff:a5:2d
HP Printer	70:5a:0f:e4:9b:c0
iHome PowerPlug	74:c6:3b:29:d7:1d
LiFX Bulb	d0:73:d5:01:83:08
NEST Smoke Sensor	18:b4:30:25:be:e4
Netatmo Camera	70:ee:50:18:34:43
Netatmo Weather station	70:ee:50:03:b8:ac
Phillip Hue Lightbulb	00:17:88:2b:9a:25
Pixstart photo frame	e0:76:d0:33:bb:85
Ring Door Bell	88:4a:ea:31:66:9d
Samsung Smart Cam	00:16:6c:ab:6b:88
Smart Things	d0:52:a8:00:67:5e
TP-Link Camera	f4:f2:6d:93:51:f1
TP-Link Plug	50:c7:bf:00:56:39
Triby Speaker	18:b7:9e:02:20:44
Withings Baby Monitor	00:24:e4:10:ee:4c
Withings Scale	00:24:e4:1b:6f:96
Withings Sleep Sensor	00:24:e4:20:28:c6

Algorithm 2 Flows Collection Algorithm

Input: dataset pcap files.
Output: JSON files.
for each pcap file as *pFileName* **do**
  Open *pFileName* for reading
  Set $deviceCo = 1$
  Set $json = pFileName + deviceCo$
  for each MAC address in Table III as *mac* **do**
    Run `joy` with *pFileName* as input, *json* as output, and *mac* as the host MAC address
    Set $deviceCo = deviceCo + 1$
  end for
  Close *pFileName*
end for

three layers, each having 25 neurons. The first two layers use the ReLu activation function, while the last layer uses a softmax activation function. Adam optimizer is utilized for optimization.

Finally, in the *data set splitting phase* (Phase-5), we create N data sets to represent local data sets for the N candidate clients. Each of the N data sets is created randomly from the training data set. To reflect a real scenario, we ensure that those data sets do not have the same size. The majority of candidate clients possess a small amount of the data set

TABLE IV
IOT DEVICE FEATURES

Feature	*Description*
$totalSleepTime$	Total time of no activity
$totalActiveTime$	Total time of activity
$totalFlowVolume$	Number of bytes (sent/received) by the IoT device
$flowRate$	Total flow volume divided by total active time
$avgPacketSize$	Number of bytes sent or received divided by no. of packets sent or received
$numberOfServers$	Number of servers Excluding DNS (53) and NTP (123)
$numberOfProtocols$	Number of protocols based on destination port number
$numberOfUniqueDNS$	Number of unique DNS requests
$DNSinterval$	Total time for using DNS
$NTPinterval$	Total time for using NTP

TABLE V
FL PARAMETERS

Parameter	*Value(s)*
Batch size	3
E (Epochs)	8
K (Communication rounds)	20
Test dataset	20% of the dataset
Train dataset	80% of the dataset
Number of fat clients	20% of N
Number of thin clients	80% of N
Fat client dataset	10% of the train dataset
Thin client dataset	1% of train dataset

while the minority of candidate clients possess large portions of the data set. Consequently, the fat clients constitute 20% of candidate clients and each fat client has about 10% of the training data set selected randomly. On the other hand, the thin clients constitute 80% of candidate clients and each thin client has about 1% of the training data set randomly selected. All these parameters along with other FL parameters are listed in Table V.

C. Experiments

After Phase-5, N candidate clients' data sets are formed to simulate a hybrid FL setting, which is utilized in the conducted experiments. To measure the performance of the proposed heuristic, we compare the results of two algorithms against the proposed heuristic. The two algorithms are the online random algorithm and the offline best algorithm. The online random algorithm selects and rejects candidate clients randomly. On the other hand, the offline best algorithm is an offline algorithm

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3 Features Extraction Algorithm

Input: JSON files.
Output: features CSV file.
Open features file for writing
Set *maxPeriod* = 10 minutes
for each JSON file **do**
  Open JSON file for reading
  Read *deviceID* from JSON file
  Set *totalSleepTime* = 0; *totalActiveTime* = 0
  Set *totalFlowVolume* = 0; *totalPackets* = 0
  Set *numberOfServers* = 0; *numberOfProtocols* = 0
  Set *numberOfUniqueDNS* = 0; *DNSinterval* = 0
  Set *NTPinterval* = 0; *lastFlowEndTime* = 0
  for each flow **do**
    Set #flow = flow number
    Set *flowTime* = *flowEndTime* − *flowStartTime*
    Set *totalActiveTime* = *totalActiveTime* + *flowTime*
    Set *totalFlowVolume* = *totalFlowVolume*+ number of bytes in the flow
    Set *totalPackets* = *totalPackets*+ number of packets in the flow
    if port in flow is not recorded before **then**
      Set *numberOfProtocols* = *numberOfProtocols* + 1
      Record port
    end if
    if port in flow = 53 **then**
      Set *DNSinterval* = *DNSinterval* + *flowTime*
      if DNS query in flow is not recorded before **then**
        Set *numberOfUniqueDNS* = *numberOfUniqueDNS* + 1
        Record DNS query
      end if
    else if port in flow = 123 **then**
      Set *NTPinterval* = *NTPinterval* + *flowTime*
    else
      if destination address in flow is not recorded before **then**
        Set *numberOfServers* = *numberOfServers* + 1
        Record destination address
      end if
    end if
    if #flow = 1 **then**
      Set *startTime* = *flowStartTime*
    else
      Set *totalSleepTime* = *totalSleepTime*+ (*flowStartTime* − *lastFlowEndTime*)
      if *flowEndTime* − *startTime* ≥ *maxPeriod* **then**
        Set *flowRate* = 0
        if *totalActiveTime* ≥ 0 **then**
          Set *flowRate* = *totalFlowVolume*/*totalActiveTime*
        end if
        Set *avgPacketSize* = 0
        if *totalPackets* ≥ 0 **then**
          Set *avgPacketSize* = *totalFlowVolume*/*totalPackets*
        end if
        Add a record to features file with features and *deviceID*
        Reinitialize all features variables
      end if
    end if
    Set *lastFlowEndTime* = *flowEndTime*
  end for
  Close JSON file
end for
Close features file

that can work with all candidate clients at the same time. In other words, the offline best algorithm does not have to wait for clients to be available over time and instead have the advantage of working with all candidate clients at the same time. The offline best algorithm creates a sorted list of all (i.e., N) candidate clients based on accuracy and selects the top R candidate clients, which are mostly fat candidate clients.

We conduct 125 experiments to test the performance of the proposed heuristic. In all these experiments, we fixed the number of communication rounds to 20; the number of epochs per client to 8; and the batch size to 3 (as shown in Table V). We are not interested in optimizing the aforementioned parameters since the goal of this article is not to achieve the highest accuracy possible, but to investigate the ability of the proposed heuristic compared against the state-of-the-art algorithms. Thus, we vary the number of clients N, the number of selected candidate clients R, and r_2 (used to compute the value of α^*) each with five different values as indicated in Table VI. Experimenting with different values of N, R, and r_2 is vital to truly test the abilities of the proposed heuristic.

TABLE VI
SIMULATION PARAMETERS

Sym.	*Parameter*	*Value(s)*
N	No. of candidate clients	(100, 200, 400, 800, and 1600)
R	No. of best candidate clients	(10, 20, 30, 40, and 50)
r_1	Minimum no. of best candidate clients	1
r_2	Maximum no. of best candidate clients	(1, 2, 3, 4, and 5)

The values in Table VI are not selected arbitrarily. We test with values of N that vary from hundreds to thousands by doubling the numbers to see how this increase affects the performance. As for R, we test with different values in tens and noticed that raising R more is not interested since the performance of all algorithms converges as explained later. Setting r_1 to one is a must since we need to select the first best candidate client. Additionally, we noticed that raising the value of r_2 to more than 5 will results in a very low α^* especially when N is 100. In other words, setting r_2 to higher numbers will reduce the search size to zero candidate clients since α^* will be close to zero.

VII. DISCUSSION

Since we cannot present all values and figures of the 125 experiments, we fixed in each case 2 of the variables (i.e., N, R, and r_2) and show the results for changing the third variable. Also, for each case, we present three figures: the first represents the general accuracy of the system; the second for the average accuracy of selected candidate clients, which is used to indicate the contribution of individual candidate clients toward the general accuracy of the system; and finally, the percentage of the accepted Fat clients, which is useful for investigating if more Fat clients lead to higher accuracy.

A. Experimenting With Different Values of r_2

The results illustrated in Fig. 5 support the discussion in Section V. Increasing the value of r_2 while fixing r_1 to 1 reduces the value of α^* and thus increases the probability of finding the best candidate clients as shown in Fig. 5(a). Furthermore, Fig. 5(a) and (b) confirms the fact that increasing the value of r_2 leads to accepting more fat candidate clients with higher accuracy.

B. Experimenting With Different Values of R

Fig. 6 shows that the proposed heuristic is more competitive when the number of selected candidate clients (i.e., R) is

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

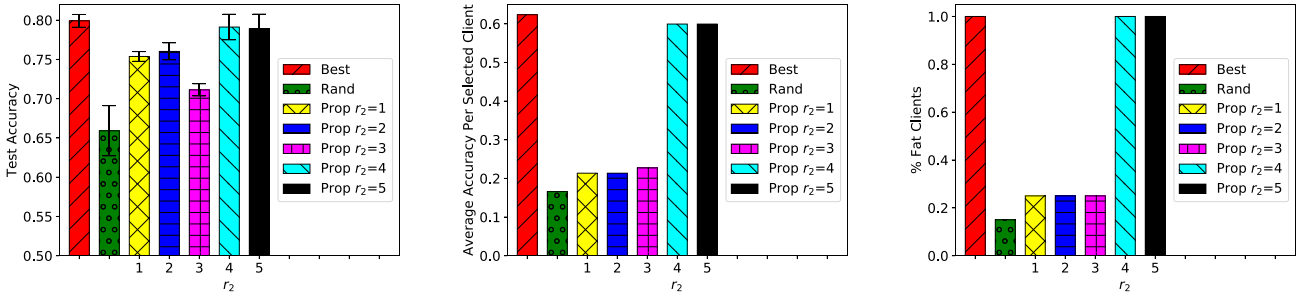

Fig. 5. Performance of algorithms for different r_2 values (1, 2, 3, 4, 5) while fixing N, number of clients, to 400 and R, number of selected clients, to 20. Our proposed algorithm performs better than the random algorithm approaching the performance of the best algorithm as r_2 is increased. (a) Test accuracy. (b) Average accuracy per selected client. (c) % Fat clients.

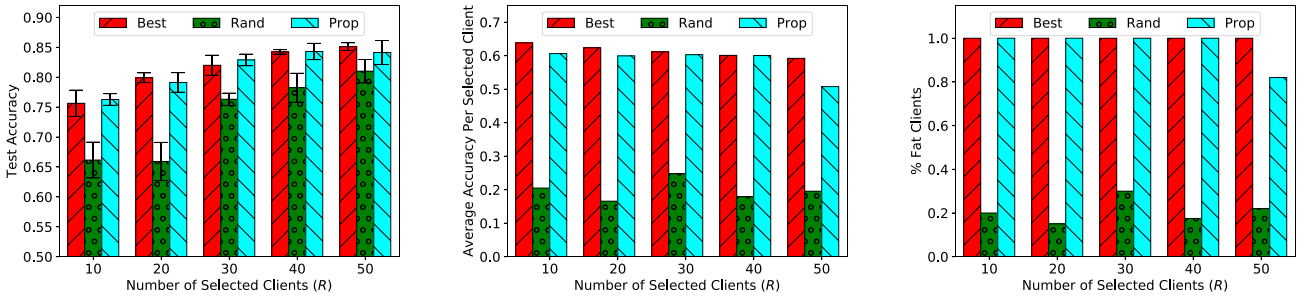

Fig. 6. Performance of algorithms for different R, number of selected clients, values (10, 20, 30, 40, 50) while fixing N, number of clients, to 400 and r_2 to 4 (α^* is 43). Our proposed algorithm is more competitive for smaller values of R and as R is increased, the performance of algorithms converges. (a) Test accuracy. (b) Average accuracy per selected client. (c) % Fat clients.

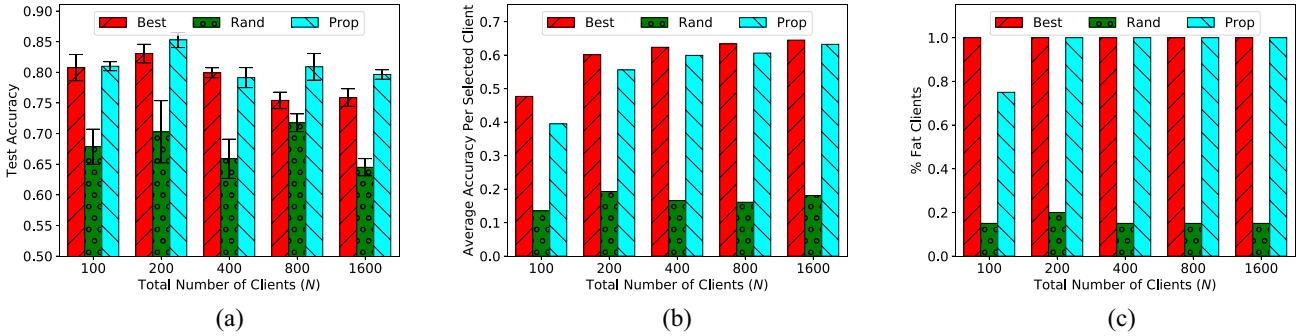

Fig. 7. Performance of algorithms for different N, number of clients, values (100, 200, 400, 800, and 1600) while fixing R, number of selected clients, to 30 and r_2 to 4 (different α per N value). Our proposed algorithm performs better than the random algorithm approaching the performance of the best algorithm regardless of the value of N. (a) Test accuracy. (b) Average accuracy per selected client. (c) % Fat clients.

low. However, as R increases, the accuracy of all algorithms converges as indicated in Fig. 6(a). As the number of selected candidate clients increases, all algorithms will have a good portion of the data set and will be able to converge to a high accuracy in less time. As a result, there is no problem to solve for high values of R. Besides, sometimes it is not feasible to contact many candidate clients since some of them are not available. Fig. 6(b) and (c) shows that the accuracy of the system increases regardless of the accuracy of the individual candidate clients and the number of fat nodes.

C. Experimenting With Different Values of N

The performance of the proposed heuristic is almost stable when the total number of candidate clients is increased while fixing R to 30 as illustrated in Fig. 7. The accuracy of the proposed heuristic is almost 80% for different values of N as shown in Fig. 7(a). Additionally, Fig. 7(b) and (c) supports this argument.

D. Lessons Learned

We can conclude here that based on the presented results.

1) The performance of the proposed online heuristic is stable regardless of the number of total clients N, as illustrated in Fig. 7.
2) The accuracy of the proposed heuristic increases as the number of selected candidate clients R increases. However, as R goes up, the performance of the proposed online heuristic, the online random algorithm, and the

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

offline best algorithm tends to converge as indicated in Fig. 6. This is because, with more candidate clients, algorithms have access to a larger portion of the overall data set.

3) As the number of best candidate clients (r_2) is increased, the performance of the proposed online heuristic is enhanced since better candidate clients are used as shown in Fig. 5.

VIII. Conclusion

In this article, the problem of optimizing accuracy in stateful FL by selecting the best candidate clients based on test accuracy is considered. Then, the problem of maximizing the probability of selecting the best candidate clients based on accuracy is formulated as a secretary problem and performance analysis is presented along with proofs. Based on the formulation, an online stateful FL heuristic is proposed to find the best candidate clients. In addition, an IoT client alarm application is proposed that utilizes the proposed heuristic along with IoT device-type classification to identify unauthorized IoT devices and alert clients. To test the efficiency of the proposed heuristic, we run many experiments using a real IoT data set and the performance of the online random algorithm and the offline best algorithm are compared against the performance of the proposed heuristic. Results show that the proposed heuristic performs better than the two state-of-the-art algorithms. Additionally, we notice the stability in the performance of the proposed heuristic compared against the performance of the other two algorithms regardless of the number of participating candidate clients. We also notice that when increasing the number of best selected candidate clients, the proposed heuristic becomes less competitive. This is because with more clients comes more data and thus the performance of algorithms converges regardless of how bad an algorithm in selecting candidate clients.

We want to emphasize a disclaimer that the proposed algorithm is not designed to work efficiently in stateless FL. Moreover, the proposed algorithm is not competitive in applications of offline nature and/or with open budget in terms of selected clients since clients' data set size and testing accuracies are known in advance and all clients can be used in the training process.

In the future, we plan to devise different variations of the secretary problem and provide performance analysis along with proofs for each. We also intend to run several experiments using a real data set to evaluate those variations and compare their performance with the performance of the proposed heuristic.

References

[1] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2020.

[2] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[3] J. Jeon, J. Kim, J. Huh, H. Kim, and S. Cho, "Overview of distributed federated learning: Research issues, challenges, and biomedical applications," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Jeju Island, South Korea, Oct. 2019, pp. 1426–1427.

[4] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[6] Q. Li, Z. Wen, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," 2019. [Online]. Available: http://arxiv.org/abs/1907.09693.

[7] D. Liu, T. Miller, R. Sayeed, and K. Mandl, "FADL:Federated-autonomous deep learning for distributed electronic health record," 2018. [Online]. Available: http://arxiv.org/abs/1811.11400.

[8] D. Conway-Jones, T. Tuor, S. Wang, and K. Leung, "Demonstration of federated learning in a resource-constrained networked environment," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Washington, DC, USA, Jun. 2019, pp. 484–486.

[9] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[11] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[12] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: http://arxiv.org/abs/1912.04977.

[13] S. Tabatabai, I. Mohammed, A. Al-Fuqaha, and J. Qadir, "Opportunistic selection of vehicular data brokers as relay nodes to the cloud," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2020, pp. 1–6.

[14] V. Selis and A. Marshall, "A classification-based algorithm to detect forged embedded machines in IoT environments," *IEEE Syst. J.*, vol. 13, no. 1, pp. 389–399, Mar. 2019.

[15] W. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," 2019. [Online]. Available: http://arxiv.org/abs/1909.11875.

[16] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019. [Online]. Available: http://arxiv.org/abs/1902.01046.

[17] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[18] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[19] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

[20] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 8, pp. 1754–1766, Aug. 2020.

[21] Y. Liu *et al.*, "A communication efficient collaborative learning framework for distributed features," 2020. [Online]. Available: http://arxiv.org/abs/1912.11187.

[22] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Taichung, Taiwan, Dec. 2018, pp. 1–4.

[23] L. Liu, J. Zhang, S. Song, and K. Letaief, "Client-edge-cloud hierarchical federated learning," 2019. [Online]. Available: http://arxiv.org/abs/1905.06641.

[24] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported Internet of Things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.

[25] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "Performance optimization of federated learning over wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[26] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–7.

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

[27] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, Jul. 2019, pp. 954–964.

[28] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1345–1348, Oct. 2019.

[29] H. Nguyen, N. Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach," 2019. [Online]. Available: http://arxiv.org/abs/1910.09172.

[30] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data," 2019. [Online]. Available: http://arxiv.org/abs/1905.07210.

[31] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[32] J. Konecný, H. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017. [Online]. Available: http://arxiv.org/abs/1610.05492.

[33] S. Caldas, J. Konecný, H. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018. [Online]. Available: http://arxiv.org/abs/1812.07210.

[34] T. Ferguson, "Who solved the secretary problem," *Stat. Sci.*, vol. 4, no. 2, pp. 282–289, 1989.

[35] L. Bayón, P. Fortuny, J. Grau, A. Oller-Marcén, and M. Ruiz, "The best-or-worst and the postdoc problems with random number of candidates," *J. Comb. Optim.*, vol. 38, pp. 86–110, Jul. 2019.

[36] P. Freeman, "The secretary problem and its extensions: A review," *Int. Stat. Rev.*, vol. 51, pp. 189–206, Aug. 1983.

[37] J. Gilbert and F. Mosteller, "Recognizing the maximum of a sequence," *J. Amer. Stat. Assoc.*, vol. 61, no. 313, pp. 35–73, 1966. [Online]. Available: https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1966.10502008

[38] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions," in *Proc. 16th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2005, pp. 630–631.

[39] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "Online auctions and generalized secretary problems," *ACM SIGecom Exchanges*, vol. 7, no. 2, p. 7, Jun. 2008. [Online]. Available: https://doi.org/10.1145/1399589.1399596

[40] B. Qolomany, K. Ahmad, A. Al-Fuqaha, and J. Qadir, "Particle swarm optimized federated learning for industrial IoT and smart city services," 2020. [Online]. Available: http://arxiv.org/abs/2009.02560.

[41] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018. [Online]. Available: http://arxiv.org/abs/1812.06127.

[42] L. Guo and I. Matta, "The war between mice and elephants," in *Proc. 9th Int. Conf. Netw. Protocols (ICNP)*, Riverside, CA, USA, Nov. 2001, pp. 180–188.

[43] N. Buchbinder, K. Jain, and M. Singh, "Secretary problems via linear programming," *Math. Oper. Res.*, vol. 39, no. 1, pp. 190–206, 2014.

[44] R. Vaze, "Competitive ratio analysis of online algorithms to minimize packet transmission time in energy harvesting communication system," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 115–1123.

[45] M. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 5187–5192.

[46] M. Santos, R. Andrade, D. Gomes, and A. Callado, "An efficient approach for device identification and traffic classification in IoT ecosystems," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Natal, Brazil, Jun. 2018, pp. 304–309.

[47] J. Bugeja, P. Davidsson, and A. Jacobsson, "Functional classification and quantitative analysis of smart connected home devices," in *Proc. Global Internet Things Summit (GIoTS)*, Bilbao, Spain, Jun. 2018, pp. 1–6.

[48] B. Desai, D. Divakaran, I. Nevat, G. Peter, and M. Gurusamy, "A feature-ranking framework for IoT device classification," in *Proc. 11th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Bengaluru, India, Jan. 2019, pp. 64–71.

[49] F. Shaikh, E. Bou-Harb, J. Crichigno, and N. Ghani, "A machine learning model for classifying unsolicited IoT devices by observing network telescopes," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Limassol, Cyprus, Jun. 2018, pp. 938–943.

[50] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 2177–2184.

[51] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, May 2017, pp. 559–564.

[52] Y. Meidan *et al.*, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 506–509. [Online]. Available: https://doi.org/10.1145/3019612.3019878

[53] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[54] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 114–129, Jan./Feb. 2020.

[55] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, early access, Sep. 1, 2020, doi: 10.1109/MNET.011.2000286.

[56] P. Perricone, B. Hudson, B. Anderson, B. Long, and D. McGrew, *Joy A Package for Capturing and Analyzing Network Data Features*, 2nd ed., Cisco Syst., San Jose, CA, USA, Jan. 2018. [Online]. Available: https://github.com/cisco/joy/blob/master/doc/using-joy-05.pdf

[57] I. Mohammed. (2020). *Federated IoT Classification*. [Online]. Available: https://github.com/IhabMoha/Federated-IoT-Classification

[58] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

Ihab Mohammed (Member, IEEE) received the B.S. and M.S. degrees in computer science from Al-Nahrain University, Baghdad, Iraq, in 2002 and 2005, respectively, and the Ph.D. degree from Western Michigan University, Kalamazoo, MI, USA, in 2020.

He is an Assistant Professor with the School of Computer Sciences, Western Illinois University, Macomb, IL, USA. In 2005, he joined the Department of Computer Science, Al-Nahrain University as a Lecturer. In 2014, he joined Western Michigan University as a Ph.D. student and worked in different positions, including a Research Assistant, a Graduate Teaching Assistant, and a Part-Time Faculty Instructor. His current research interests include Internet of Things, security, cloud computing, data analysis and algorithm design with machine learning, deep learning, and federated learning.

Dr. Mohammed served as a Reviewer for high-quality journals, such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE INTERNET OF THINGS JOURNAL, *IET Networks*, and *IET Intelligent Transport Systems*. Additionally, he served as a Reviewer for several conferences, including IEEE VTC-2018, IEEE ISCC 2018, and IEEE IWCMC 2018.

Shadha Tabatabai (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in computer science from Al-Nahrain University, Baghdad, Iraq, in 1998 and 2001, respectively. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Western Michigan University (WMU), Kalamazoo, MI, USA.

She joined the Department of Computer Science, Al-Nahrain University in 2002 as an Assistant Lecturer and promoted to a Lecturer in 2012. She also served as a Research Assistant, a Graduate Teaching Assistant, and a part-time Faculty Instructor with the Department of Computer Science, WMU. Her current research interests include Internet of Things, cloud computing, machine learning, and reinforcement learning.

Ms. Tabatabai served as a reviewer for different conferences, including IEEE ISCC 2018, IEEE IWCMC 2018, and IEEE VTC-2018.

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.

Ala Al-Fuqaha (Senior Member, IEEE) received the Ph.D. degree in computer engineering and networking from the University of Missouri–Kansas City, Kansas City, MO, USA, in 2004.

He is currently a Professor with the Information and Computing Technology Division, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar, and the Computer Science Department, Western Michigan University, Kalamazoo, MI, USA. His research interests include the use of machine learning in general and deep learning in particular in support of the data-driven and self-driven management of large-scale deployments of IoT and smart city infrastructure and services, wireless vehicular networks, cooperation and spectrum access etiquette in cognitive radio networks, and management and planning of software-defined networks.

Prof. Al-Fuqaha serves on the editorial boards of multiple journals, including IEEE COMMUNICATIONS LETTER and *IEEE Network Magazine*. He also served as the Chair, the Co-Chair, and a Technical Program Committee Member of multiple international conferences, including IEEE VTC, IEEE Globecom, IEEE ICC, and IWCMC. He is an ABET Program Evaluator.

Faissal El Bouanani (Senior Member, IEEE) was born in Nador, Morocco, in 1974. He received the M.S. and Ph.D. degrees in network and communication engineering from Mohammed V University-Souissi, Rabat, Morocco, in 2004 and 2009, respectively.

He has served as a Faculty Member with the University of Moulay Ismail, Meknes, Morocco, from 1997 to 2009, before joining the National High School of IT/ENSIAS College of Engineering, Mohammed V University, Rabat, in 2009, where he is currently an Associate Professor. He advised many Ph.D. and master's students with Mohammed V University and University of Moulay Ismail. His research efforts have culminated in more than 75 papers in a wide variety of international conferences and journals. His current research interests include performance analysis and design of wireless communication systems.

Dr. El Bouanani was awarded the best one by Mohammed V University-Souissi for his Ph.D. thesis in 2010. He served as the TPC Chair of the ICSDE conferences and the General Co-Chair of ACOSIS'16 and CommNet'18 conferences. He serves as the General Chair of the 2019/2020 CommNet conferences. He has also been involved as a TPC member in various conferences and IEEE journals. He is also an Associate Editor of IEEE ACCESS and an Editor of *Frontiers in Communications and Networks*.

Junaid Qadir (Senior Member, IEEE) received the bachelor's degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2000, and the Ph.D. degree from the University of New South Wales, Kensington, NSW, Australia, in 2008.

He is a Professor with the Information Technology University (ITU), Lahore, where he also serves as the Chairperson of the Electrical Engineering Department. He is the Director of the IHSAN Research Lab, ITU, since December 2015. He has published more than 100 peer-reviewed articles at various high-quality research venues, including more than 50 impact-factor journal publications at top international research journals, including *IEEE Communication Magazine*, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and IEEE TRANSACTIONS ON MOBILE COMPUTING. His primary research interests are in the areas of computer systems and networking, applied machine learning, using ICT for development, and engineering education.

Prof. Qadir is an Award-Winning Teacher who has been awarded the Highest National Teaching Award in Pakistan—the Higher Education Commission's Best University Teacher Award—for the year 2012–2013. He has been an ACM Distinguished Speaker since January 2020. He is a Senior Member of ACM.

Basheer Qolomany (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the University of Mosul, Mosul City, Iraq, in 2008 and 2011, respectively, and the Ph.D. and second master's en-route to Ph.D. degrees in computer science from Western Michigan University (WMU), Kalamazoo, MI, USA, in 2018.

He is currently an Assistant Professor with the Department of Cyber Systems, University of Nebraska at Kearney, Kearney, NE, USA. He served as a Visiting Assistant Professor with the Department of Computer Science, Kennesaw State University, Marietta, GA, USA, in 2018 and 2019, and a Graduate Doctoral Assistant with the Department of Computer Science, WMU, from 2016 to 2018. He also served as a Lecturer with the Department of Computer Science, University of Duhok, Dahuk, Iraq, from 2011 to 2013. His research interests include machine learning, deep learning, Internet of Things, smart services, cloud computing, and big data analytics.

Dr. Qolomany has served as a reviewer of multiple journals, including IEEE INTERNET OF THINGS JOURNAL, *Energies—Open Access Journal*, and *Computers and Electrical Engineering* (Elsevier). He also served as a Technical Program Committee Member and a reviewer of some international conferences, including IEEE Globecom, IEEE IWCMC, and IEEE VTC.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor with the Computer Science and Engineering Department, Qatar University, Doha, Qatar. Previously, he served in different academic and administrative positions with the University of Idaho, Moscow, ID, USA, Western Michigan University, Kalamazoo, MI, USA, University of West Florida, Pensacola, FL, USA, University of Missouri–Kansas City, Kansas City, MO, USA, University of Colorado–Boulder, Boulder, CO, USA, and Syracuse University. He has authored nine books and more than 600 publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid.

Prof. Guizani is the recipient of the 2017 IEEE Communications Society Wireless Technical Committee Recognition Award, the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks and the 2019 IEEE Communications and Information Security Technical Recognition Award for outstanding contributions to the technological advancement of security. Throughout his career, he received three teaching awards and four research awards. He is currently the Editor-in-Chief of the *IEEE Network Magazine*, serves on the editorial boards of several international technical journals and the Founder and Editor-in-Chief of *Wireless Communications* and *Mobile Computing* journal (Wiley). He guest edited a number of special issues in IEEE journals and magazines. He also served as a member, a chair, and a general chair of a number of international conferences. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. He is a Senior Member of ACM.

Authorized licensed use limited to: Qatar University. Downloaded on November 06,2022 at 11:41:31 UTC from IEEE Xplore. Restrictions apply.