IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 September 2023, accepted 8 October 2023, date of publication 13 October 2023, date of current version 18 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324178

RESEARCH ARTICLE

Trust-Aware Scheduling for Edge Computing With Task Dependencies and Unreliable Servers

YOUSEF ALSENANI AND ABDULAZIZ S. ALNORI
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
Corresponding author: Yousef Alsenani (yalsenani@kau.edu.sa)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant G: 689-611-1443.

ABSTRACT Volunteer Edge Computing (VEC) is a promising solution for addressing the challenge of high round-trip latency in traditional cloud computing systems. Leveraging distributed computing resources reduces latency and improves performance. However, resource management in VEC is challenging, first, due to the uncertain behavior of volunteers, which frequently go offline unexpectedly, and second, since sequences of tasks can be executed on different volunteers, which requires transmitting data from one to another volunteer, which can lead to processing interruptions and network overhead. To address these challenges, we propose a trust-aware scheduling procedure that consists of two stages. First, we train a regression model based on lagged data suitable to accurately predict volunteer availability. Second, we assign tasks to volunteers using a metric based on the predicted availability from the first stage. The metric assesses the likelihood that a candidate volunteer can successfully complete a task and the likelihood that nearby nodes are available for successor tasks or as replacements if the processing is not completed. Thereby, we increase the chances of assigning tasks with dependencies to nearby resources, thus reducing long-distance communication and hence latency. We evaluate our approach in a discrete-event simulation using real data from Telecom's base stations. The simulation results indicate significant improvements in task failures, task completion rates, delays, and average execution times when compared to the existing alternative algorithm.

INDEX TERMS Edge computing, volunteer edge computing, trust, availability, scheduling, task dependencies.

I. INTRODUCTION

Volunteer computing is a concept for distributed computing where servers voluntarily offer their computers' spare resources to a certain project for computing purposes [1], [2]. One of the most well-known volunteer computing projects is Folding@home[1], which assembles computing resources for medical applications. Initially, its management stated that the project could call upon 470 petaFLOPS of raw computing power for COVID-19 research [3]. Compared to current high-end desktop PCs, this is about 500,000 times quicker. Volunteers can also conveniently be integrated into edge computing since it is likely that some volunteers are located close to the nodes where demand originates [4], [5]. This new paradigm is called volunteer edge computing. While in edge computing, users offload their tasks to the closest dedicated edge server, in volunteer edge computing, users offload their tasks to nearby volunteers, which come from various sources and differ significantly in terms of computing power, duration of availability, reliability, and trustworthiness. Therefore, we generally expect a network of volunteers to be more distributed and heterogeneous than a network of edge servers. Moreover, since volunteers offer spare computing resources of less powerful computers, their overall computing capability may not suffice to handle resource-intensive applications. As a result, volunteers can complement dedicated edge servers but not replace them [6], [7]. Managing the network of computing resources is crucial for effectively utilizing their computing power [8], [9].

A frequent issue encountered in volunteer computing is task failures, that is, tasks that are not processed because

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Yan.

[1]https://foldingathome.org/

the corresponding volunteer goes offline unexpectedly before completing the computations, in which case, all progress is lost. Recently, several studies have proposed approaches to enhance system performance and decrease task failure rates in edge computing environments [6], [10], [11]. However, existing approaches do not consider task dependencies. In the presence of task dependencies, the output of one task is the input for the next task. Hence, in case two dependent tasks are not executed on the same machine, the data is forwarded from one volunteer to the next. If the distance between these two volunteers is long, latency and network overhead increases.

Our approach, on the other hand, explicitly considers the availability of each feasible volunteer and its nearby volunteers to increase the likelihood of (i) assigning tasks to volunteers that finish processing, and (ii) assigning tasks with dependencies to volunteers where other volunteers are readily available in close proximity. As a result, our approach reduces task failures and long-distance communication which results in reduced task completion times, especially when processing sequences of dependent tasks. The approach comprises two phases. In the first phase, we train a machine learning model based on lagged data to predict volunteer availability. The model can be trained ahead of time, i.e, offline. The second phase concerns the assignment of tasks to volunteers, online, as task arrive. Hence, the computational time should be as close to realtime as possible to not introduce additional delays. Therefore, we use a heuristic assignment procedure that makes decisions quickly and intelligently based on the predictions that are made with the model from Phase 1. This assignment logic uses a scoring scheme, where the variables to calculate the scores are predicted by a machine learning model. The scoring metric combines two aspects: (i) the likelihood that the candidate volunteer under consideration can complete the task and (ii) the likelihood that other nodes close by can support or replace the candidate volunteer should it be required. We call this score the trust of the volunteer. We then rank the feasible candidates and assign the task at hand to the top candidate.

The main contributions of this study are the following.

- A regression model based on lagged data to predict volunteer availability.
- A trust metric that considers the availability of individual and nearby volunteers.
- A simulation model to assess the performance of our approach and compare it to alternative resource management systems common in VEC.
- Management insights derived from our computational study based on Telecom's base station dataset.

The remainder of the paper is organized as follows. Section II gives on overview on the related literature. We describe the problem in detail in Section III and develop our solution approach in Section IV. Then, we discuss the experimental results in Section V. Then, in Section VI we briefly discuss the limitations of this work and future work. Finally, we conclude in Section VII.

II. RELATED WORK

In computing, trust remains one of the main issues in centralized and distributed systems such as cloud, edge [9], [12], [13], volunteer computing [14], [15], and federated learning [16], [17]. This section review number of trust research in volunteer computing and other emerging paradigms, such as volunteer edge computing, edge computing, and federated learning.

A. RESEARCH IN DISTRIBUTED SYSTEMS

Resource management is a crucial area of research in edge computing. In general, computational (e.g., CPU cycles), communication (e.g., bandwidth, power), and storage resources are the critical resource allocation resources used in the current study. The emerging paradigms of edge computing and federated learning are fit for latency-sensitive and private applications. The computation is processed either at the edge network or on the client's side. Our model shares a similar concept to Rjoub et al. in [18] recommending that establishing trust between the edge server and edge nodes should be a fundamental component of the decision-making procedure. During the local training, the trust model sought to identify the resource that was either overused or underused. Then, the authors provided DDQN-Trust, a selection method based on double deep Q learning that considers the edge nodes' energy levels and trusts ratings to determine the best scheduling options. The model was then integrated into four federated learning aggregation techniques. A real-world dataset used in experiments demonstrates that the DDQN Trust approach consistently outperforms existing models. In contrast to this approach, our approach relies on the availability of nodes.

The authors in [19] proposed an effective and trustworthy system based on edge computing and blockchain. Initially, a group technique with a trust system level was intended to increase transmission effectiveness and guarantee the trustworthiness of edge nodes during interactions. Then, a stacked task is proposed, a sorting and ranking system that categorizes and orders processing jobs based on their importance. The most important job may be executed in the address with the resources by matching the task stack and address stack. As a result, the job failure ratio can be significantly decreased. Finally, the authors used a productive and reliable content approach. It used Zipf distribution to identify terms with high content popularity while uploading hot data's ciphertext to the cloud server after utilizing SSE and the associated index to the blockchain, making it simple for all users to correctly and quickly search data.

A reputation schema is proposed in the new paradigm of federated learning. Machine learning is employed on the mobile device instead of a centralized server to decrease privacy and bandwidth issues [20]. The authors proposed a schema to evaluate a mobile device called a worker to handle any unreliable data resulting from a malicious or untrusted user. A distributed strategy called Consortium blockchain is

used to effectively maintain device profiles without altering or repudiation. The proposed model improved the reliability of mobile federated learning. Although their approach delivers a reliable network, it might suffer from hard-deadline tasks.

The authors in [21] outline the essential characteristics of federated learning on edge computing, such as customization, decentralization, reward systems, trust, activity tracking, diversity and context awareness, and bandwidth. Additionally, they introduce the reputation-aware FL based on blockchain to guarantee reliable federated learning in edge infrastructure. Xiaoyu et al. presented FLTrust [22] to accomplish Byzantine stability versus untrusted nodes. FLTrust is a Federated learning approach for prior knowledge trust. The service provider gathers training data for the learning task and constructs a model to generate trust. The service provider gives the users a trust rate from each local model update at each iteration. Users will receive a lower trust rate if their model diverges from the global model. We believe thier approach is more suitable to the federated learning paradigm, while our approach can be applied well to edge computing and federated learning environment.

An ideal job scheduling strategy for mobile nodes is presented, focusing on the dynamic voltage and frequency scaling and whale optimization methods [23]. This research provided a trade-off between effectiveness and power usage by tackling the joint optimization for task completion time and energy consumption at the same time. It does this by taking into account different components: task execution place, task execution pattern, and running voltage and frequency of mobile nodes. The method has distinctive efficiency and operational cost, offering a practical approach to comparable optimization challenges of cloud applications, according to a series of exhaustive modeling findings.

The task allocation to users in mobile systems is one of the major concerns. The author used a reliable matching concept to facilitate an acceptable pairing between different groups of entities based on interests to overcome this limitation [24]. As a result, they start by defining two distinct stability criteria for user satisfaction in MCS platforms. Then, they explore the stability assurances of various effective, robust task allocation techniques in distinct MCS cases. Finally, we demonstrate that the suggested algorithms outperform the state-of-the-art approaches by thoroughly simulating their efficiency on a real dataset. The authors in [25] proposed a light trust model that asset the trust of edge node and managed SLA. Trust is evaluated through the quality of service parameter services such as latency, packet loss percentage, jitter, throughput, and task failure rate. Our approach evaluates the trust from a resource and host-based perspective while The authors in [25] consider a network behavior in their evaluation.

B. RESEARCH IN VOLUNTEER COMPUTING

The authors in [6] proposed ProTrust probabilistic trust framework in volunteer cloud computing. The framework proposed two terms; trustworthiness based on the priority of a task, and (2) trustworthiness computed by behavior detection. ProTrust used an enhanced Beta function and behavioral change detection in the trust computations. The volunteer computing project in [26] developed a web-based platform for volunteer computing for hydrological applications. The platform only required a web browser for the computation. The platform offers scaling and distribution capabilities for projects with a large number of volunteer resources. The project used a large-scale hydrological flood forecasting model as a case study to test and assess the suggested methodology. However, all participants in this platform are assumed to be trusted.

The authors in [27] proposed a trust-aware in social IoT, and the model selected a candidate node based on friendship trust in the edge system to offer message service. The approach shows promising results for the feasibility of stable communication. However,the QoS requirements are not analyzed in the system. Trust evaluation should consider the node's availability to ensure satisfactory performance.

The authors in [28] proposed a task coordinating and job scheduling techniques in Vehicular Ad hoc Networks (VANETs). The authors developed an adaptive task replication approach to provide fault tolerance by preventing task failures due to object location. The authors in [29] presented a replication mechanism for Micro-Blogging Service run over in volunteer computing infrastructure. While replicated Blogs enhance the reliability of the service, the authors finally analyze the trade-off between blog replica availability and efficiency. The authors in [30] proposed a novel availability model in large volunteer computing. The model clusters a large number of volunteer participants with low availability profiles to host a service. The group is equivalent to one dedicated server. The authors in [31] present a fog computing model based on volunteer computing to minimize network delay and energy power. They consider the performance metrics' delay, power consumption, and network usage. The author evaluated the model through an iFogSim simulator, presenting a significant reduction.

The contributions, limitations, and justifications with the current methods is summarized in Table1. Most existing trust models evaluate the resource based on behavior and feedback and others based on power and energy capabilities. Other works consider the quality of service QoS to assess the network of edge nodes. We draw a conclusion to this part by outlining the problems we found in the related work. Trust evaluation should proactively consider the node's availability to improve system reliability, mitigate expected failures or misbehavior, and account for task dependencies to ensure satisfactory performance in the edge computing environment. Therefore, we propose a machine learning trust framework specifically designed for task dependencies in volunteer edge computing.

III. PROBLEM DESCRIPTION AND FORMULATION

In this section, we describe our problem in detail covering the system's architecture, the process of communication,

TABLE 1. Contributions, limitations, and justifications with the current methods.

Ref.	Contributions	Limitations	Justifications
[32]	Employment of subjective logic to gauge trust level.	Incorporation of multiple layers demands increased computational resources.	interdependencies between tasks are not considered
[33], [34]	Use statistical functions like the beta function to assess trust levels.	The model doesn't update to reflect recent behavioral changes.	Our framework uses the ML model to have an accurate result
[35]	Incorporation of a feedback-containing database for nodes.	Decentralization of databases may result in integrity issues.	The trust model does not consider task dependencies.
[28]	Developed an adaptive task replication approach to provide fault tolerance in (VANET)	Task replication might cause overhead	Our approach rely more on proactive technique
[25]	Trust is evaluated through the quality of service parameter services such as latency, packet loss percentage, etc.	The model considers only a network behavior in their evaluation.	Our approach is more adaptive to user application performance SLAs rather than network QoS.
[6]	The work used a statistical approach to evaluate change behavior and trust.	There is no variety of reliability in the environment, and all resources are unreliable	The work is adaptable to task priority rather than dependencies.
[27]	The work selected a candidate node based on friendship trust in the edge system to offer message service	The QoS requirements are not analyzed in the system.	Trust evaluation should consider the node's availability to ensure satisfactory performance.
[36]	Integration of MRC and SC to sustain reliability.	Transmission of trust computations across multiple layers might lead to integrity issues.	interdependencies between tasks are not considered

and the life cycles of tasks and volunteer resources. Additionally, we formally describe the problem using a Mixed Integer Linear Program (MILP).

A. PROBLEM DESCRIPTION

In this section, we describe the architecture of the system in volunteer edge computing and the system's dynamics which is visualized in Figure1. There are four different types of actors participating in volunteer edge computing: base stations, edge servers, volunteers, and users. *Base stations* are part of the communication infrastructure and the connection points for end users' devices, edge servers, and volunteers alike. They connect endpoints (users and volunteers) to the rest of the network and make sure that they can communicate with the nearest edge server. *Edge servers* are dedicated management nodes that are responsible for the resource management. They keep track of all the registered volunteers in their region and assign tasks to volunteers. *Volunteers* are the nodes that offer computing resources, and the *users* are the devices that submit their tasks to the network. The users and volunteers are depicted with different icons in the dotted box in Figure 1 illustrating their heterogeneity. When a user submits a task to the network (arrow 1), it is transferred to the nearest edge server. Once the edge server has assigned the task to a volunteer, the task is sent to the corresponding

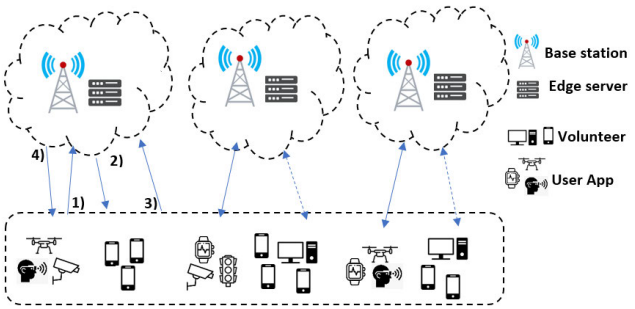

FIGURE 1. High-level architecture of a volunteer edge computing.

volunteer (arrow 2). Once processed, the volunteer server sends the results back to the edge server (arrow 3), which in turn forwards them to the requesting user (arrow 4).

Figure 2 illustrates the life cycle of tasks as a state transition diagram. Whenever a user submits a task, its base station directs the request and the data to the nearest edge server. Which then assigns the task to an available volunteer in his region. As the duration until a volunteer goes offline is uncertain, tasks are frequently allocated to volunteers that go offline before having completed processing of their current task. In such cases, the edge server will try to find another volunteer. Each task has a deadline though, and if the task is not processed before the deadline is reached, the task is

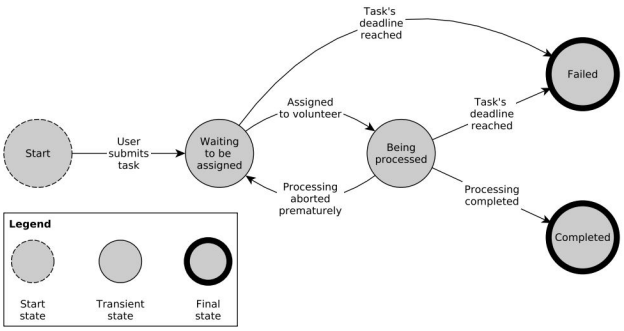

FIGURE 2. Lifecylce of tasks.

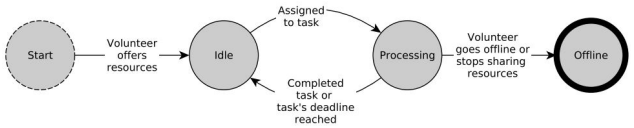

FIGURE 3. Lifecylce of volunteers.

considered failed. When task is completed the result are sent to the edge which then forwards them to the user.

Figure 3 illustrates the life cycle of volunteers. When a volunteer starts to offer computing resources, it sends a message to the nearest edge server. The edge servers then considers the volunteer for tasks that are waiting in queue and further incoming tasks. When the volunteer receives a task, it starts processing it until the task is completed, or until the task's deadline is reached, or until the volunteer goes offline. If the task is completed successfully or the task' deadline is reached the volunteer becomes idle and waits for further tasks. If the volunteer goes offline, it transitions into its terminal state.

Note that since both users and volunteers communicate with their nearest edge server and since edge servers do not forward tasks to other edger servers, we are looking at independent sub-systems — one for each edge server. For simplicity, we assume that a volunteer can only process a single task at a time. Since all progress is lost in case a volunteer does not complete processing of a task, the total accumulated yield loss is significant. An intelligent assignment logic can therefore help to reduce the *task failure rate* which is the percentage of all tasks that have failed. The complete notation is summarized in Table2

B. PROBLEM FORMULATION

In this section, we provide a Mixed Integer Linear Programming (MILP) formulation aimed at maximizing the number of tasks successfully completed by volunteers before their respective deadlines. (MILP) is recognized as a flexible and powerful methodology, especially adept at addressing problems in large-scale systems.We formulate the model for a set of tasks and a set of volunteers. Thereby, the model is sufficiently general so that we can use it for both situations that arise: (i) assigning a single task to one of multiple servers, and (ii) assigning one out of many tasks to a single idle server. Binary decision variable $x_{t,j}$ is 1 if task t is assigned

TABLE 2. Notation.

Notation	Definition
V	Set of volunteer nodes
D_{ij}	Distance between the two volunteers i and $j \in V$
$\bar{D}$	Threshold on distance for a pair of volunteers to be considered neighbors
V_j	Set of neighbors of volunteer $j \in V$
T	Set of tasks
S_j^t	Likelihood that node $j \in V$ can complete task $t \in T$
D_j^t	Distance between task $t \in T$ and volunteer $j \in V$
$\bar{D}'$	Threshold on distance for assigning a task to a volunteer
V^t	Subset of volunteers that are sufficiently close to $t \in T$, so that t can be assigned to them
DL^t	Deadline of task $t \in T$
Y_j	Random variable that denotes the duration until node j goes offline

to volunteer j, and 0 otherwise. Let $S_{t,j}$ denote the probability that volunteer j can finalize task t before the impending deadline.

$$\textit{Objective}: \quad \max \sum_{t \in \text{Tasks}} \sum_{j \in \text{Volunteers}} x_{t,j} \cdot S_{t,j} \tag{1}$$

$$\textit{Constraints}: \tag{2}$$

$$\sum_{j \in \text{Volunteers}} x_{t,j} = 1, \quad \forall t \in \text{Tasks} \tag{3}$$

$$\sum_{t \in \text{Tasks}} x_{t,j} \leq 1, \quad \forall j \in \text{Volunteers} \tag{4}$$

$$x_{t,j} \in \{0, 1\} \tag{5}$$

The objective function (1) aims to maximize the expected number of successfully finished tasks. Constraint (3) ensures every task is allocated, while constraint (4) guarantees that each volunteer is designated at most one task. Lastly, constraint (5) defines the variable domains.

IV. SOLUTION APPROACH

Our trust-aware scheduling process is divided into two stages as it shows in Figure 4. In the first phase, we train a machine learning model to predict volunteer availability using lagged data. The model can be trained ahead of time, i.e. offline. The second phase involves the online assignment of tasks to volunteers. In this phase, we utilize the model trained in Phase 1 to predict availability. With this prediction, we calculate the trust metric for each feasible volunteer, taking into account both the availability of the candidate and their vicinity. We then rank the volunteers by the trust metric and assign the task to the best volunteer.

The procedure considers the likelihood that the respective volunteer can complete the task and, additionally, the likelihood that there are other volunteers available nearby to which the task can be offloaded, should it be required. To that end, our procedure uses a scoring scheme to combine these two aspects in a single value. Henceforth, we refer to this value as *trust* or *trust score*. When a task arrives at an edge server, the server calculates the trust for each available volunteer that is within a given range of the origin of the

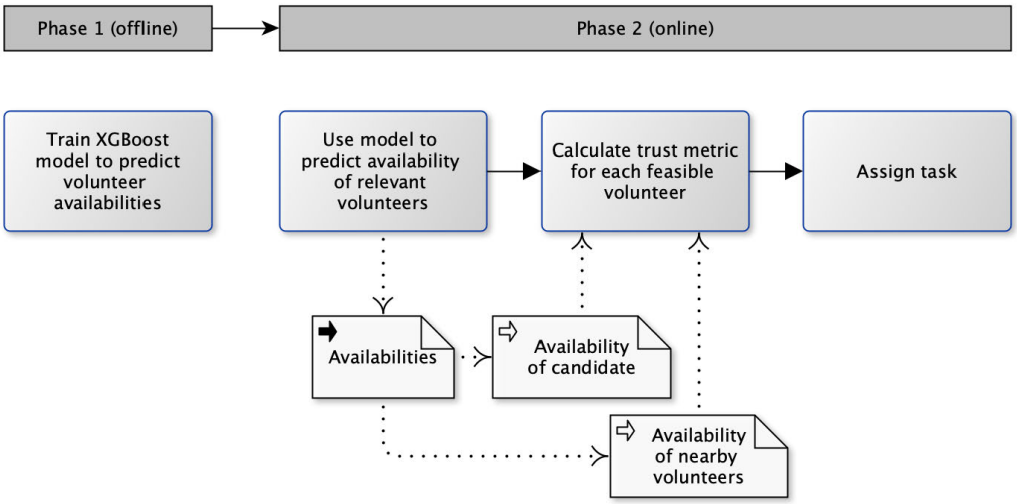

FIGURE 4. Trust-aware scheduling.

task and assigns the task to the volunteer with the highest trust score. Similarly, when a volunteer becomes available (initially offering resources or becoming idle) the server calculates its trust score for all task waiting in queue and assigns the task where the score is highest. Several variables are needed to calculate the trust score. Since these variables are uncertain, a machine learning model is employed to predict the values of these variables. Next, we introduce the trust metric and the required notation.

Let V denote the set of volunteer nodes, and let D_{ij} denote the distance between the two volunteers i and j. Two volunteers i and $j \in V$ are considered neighbors, if their distance D_{ij} is less than or equal to threshold $\bar{D}$. The set of neighbors of volunteer $j \in V$ is $V_j = \left\{i \in V \mid D_{ij} \leq \bar{D}, i \neq j\right\}$. Let T denote the set of tasks, and let D_j^t denote the distance between the origin of task $t \in T$ and volunteer $j \in V$. We can only assign a task to a volunteer, if their distance is at most $\bar{D}'$. The volunteers that are considered for task $t \in T$ is $V^t = \left\{j \in V \mid D_j^t \leq \bar{D}'\right\}$. Furthermore, let S_j^t denote the likelihood that node $j \in V$ can complete task $t \in T$. The trust for task $t \in T$ and $j \in V^t$ is calculated as

$$\mathbb{T}(t, j) = (1 - \alpha) S_j^t + \alpha \sum_{i \in V_j} S_i^t, \tag{6}$$

where α is a parameter that can be adjusted to find a good balance between the two components. Hence, $\mathbb{T}(t, j)$ can be though of as a weighted average between the likelihood that volunteer j can complete task t and the sum of the likelihoods that a neighbor can complete task t.

A. AVAILABILITY OF VOLUNTEER EDGE NODE

Next we discuss how we estimate the likelihood that node $j \in V$ can complete task $t \in T$, S_j^t. Let τ be the current time. Let random variable Y_j denote the duration until node j goes offline, and let DL^t denote the deadline of task t. We can think of S_j^t as the probability that the volunteer stays online until the task's deadline or longer, $Pr\left[DL^t \leq \tau + Y\right]$.

To calculate S_j^t, random variable Y_j is predicted with a regression model. We employ Extreme Gradient Boosting (XGBoost) for the availability estimation. XGBoost is a tree based gradient boosting algorithm that was created by Chen and Guestrin in 2016 [37]. XGBoost aims to find a relationship between the vector of input variables x_j and the output variable y_j, and it used K additive functions for the estimation as follows.

$$\hat{y}_j = \sum_{k=1}^{K} f_k\left(x_j\right). \tag{7}$$

A variation of the loss function is used to control the complexity of the trees for each set $\{(x_j, y_j)\}$, with a differentiable loss function $\mathcal{L}(\hat{y}_j, y_j)$,:

$$\mathcal{L} = \sum_j \mathcal{L}(\hat{y}_j, y_j) + \sum_k \Omega(f_k) \tag{8}$$

$$\Omega(f) = \gamma R + \frac{1}{2}\lambda ||w||^2 \tag{9}$$

Here $\mathcal{L}$ is a differentiable convex loss function that measures the difference between between the prediction and the true value. The model's complexity is penalized by the second term. Where w are the leaf output scores and R are the number of leaves on the tree, respectively. The split criterion of decision trees can incorporate this loss function, creating a pre-pruning method. Trees with higher *gamma* values are simpler. The minimum loss reduction gain required to separate an internal node is controlled by the value of *gamma*. Shrinkage, a further regularization parameter in XGBoost, lowers the step size in the additive expansion. Finally, additional methods, such as reducing the depth, etc., can also be used to reduce the complexity of the trees. The models may be trained more quickly and with less storage space because of the reduction in tree complexity.

In the feature engineering stage, we use lag-based and windows-based features. In lag-based features, we measure the availability difference in relation to prior behavior and base the model on it. More features are added to the lags, such as date, start time, latitude, longitude, user id, new start in a minute, new in start in hours, and diff (e.g., the difference between two pairs of behavior). Finally, the random variable Y_j mentioned above as the target feature for volunteer node j.

On the other hand, We apply windows-based features to compare the performance with lag-based features. Then, windows-based features are used to predict the random variable Y_j given a fixed number of previous node availability.. We used the entire dataset for training, validation, and testing. The training and testing are split into 75%, 25% respectively. We set the the depth of the model with size to eights, and assign 0.1 for the learning rate.

B. COMPLEXITY ANALYSIS OF TRUST CALCULATION AND VOLUNTEER SELECTION

In this section, we introduce the trust calculation and volunteer selection algorithm. Subsequently, we delve into the complexity analysis of our approach.

1) TRUST CALCULATION AND VOLUNTEER SELECTION

To avoid delays, it is essential that the assignment procedure delivers assignments in as close to real time as possible. In this section we show that our heuristic has a polynomial run-time which makes it suitable and scalable for online decision-making. Algorithm 1 outlines the procedure that assigns a single task to a volunteer. As mentioned, the regression model is trained beforehand in Phase 1 of our approach, so that we can use its predictions during the assignment heuristic. When a task arrives, we first calculate the trust metric for every volunteer within the specified range. Let X be the current volunteer. First, we calculate the availability of X and store the result in the trust metric (Line 6). To that end, we call the prediction function `XGBoost.predict` (Line 9). This happens in constant time. Every initial computation of a volunteer's availability gets cached (Lines 11). This strategic caching minimizes repetitive computations, permitting us to recall the precomputed value when requisite, fostering rapidity in the assignment process. Subsequently, we shift our focus to X's neighbors, deciphering their availability (Line 15-21). This process amplifies the trust value. Consequently, the availability is calculated $O(N^2)$ times. Once we have computed all trust values we can sort the volunteers with the merge sort algorithm in $O(n \log n)$ (Line 26), and finally select the first one. In total, we have a complexity of $O(N^2)$.

Algorithm 1 Calculating Trust for Node j and Its Neighbors and Selecting the Best Node

Input : Selected features for each volunteer edge node
$\bar{D}$,$\bar{D}'$ = Distance thresholds.
α = Balance thresholds
V = Set of volunteer edge nodes, t is the user task
Output : The best volunteer edge node based on trust

1 //Define necessary variables
2 τ = Current time
3 DL^t = Deadline of task t
4 **Initialize** a cache dictionary $S\_cache = \{\}$
5 //for each volunteer node within a given range
6 **foreach** $\left\{j \in V \mid D_j^t \leq \bar{D}'\right\}$ **do**
7 //Calculate the likelihood that node j can complete task t
8 **if** j *not in* S_cache **then**
9 $Y_j = XGBoost.predict(j'sdatafeatures)$
10 $S_j^t = Pr\left[DL^t \leq \tau + Y_j\right]$
11 Add j and S_j^t to S_cache
12 **end**
13 Else $S_j^t = S\_cache[j]$
14 //Calculate the availability of nearby nodes
15 **foreach** $\{i \in V | D_{ij} \leq \bar{D}\}$ **do**
16 **if** i *not in* S_cache **then**
17 $Y_i = XGBoost.predict(i'sdatafeatures)$
18 $S_i^t = Pr\left[DL^t \leq \tau + Y_i\right]$
19 Add i and S_i^t to S_cache
20 **end**
21 Else $S_i^t = S\_cache[i]$
22 **end**
23 //Calculate the trust
24 $\mathbb{T}(t,j) = (1-\alpha)\, S_j^t + \alpha \sum_{i \in V_j} S_i^t$
25 **end**
26 //Sort nodes based on trust scores and select best node
27 Sort $\mathbb{T}(t,j)$ in descending order
28 $best\_node$ = First element of sorted $\mathbb{T}(t,j)$
29 Return $best\_node$

2) COMPLEXITY ANALYSIS

In this section, we explore the complexity of our proposed approach by providing a theorem-based proof. Following that, we demonstrate the scalability of our work.

Theorem 1: For any algorithm with a polynomial run-time of $O(n^k)$, where n is the size of the input and $k \in \mathbb{N}$ is the degree of the polynomial, the algorithm demonstrates practical scalability for increasing values of n when k is relatively small.

Proof: Firstly, for comparison purposes, let's consider another algorithm with exponential run-time, $O(2^n)$. When analyzing the rate of growth, a polynomial growth for an algorithm of $O(n^k)$, with an increase from n to $n+1$, results in a change in run-time of approximately $(n+\Delta n)^k - n^k$. On the other hand, an exponential algorithm with the same increase in n would have its run-time increased by $2^{n+\Delta n} - 2^n$. Given thees observations, it becomes evident that for small values of k, the growth rate of a polynomial run-time algorithm is more contained than that of an exponential one. As a result, the scalability remains practical as the input size n grows, especially when k is a small constant. ■

TABLE 3. Data set description.

Parameter Name	Description
Month	The month when one record happens
Date	The date when one record happens
Start Time	The time when an edge node joins the edge cloud
End Time	The time when an edge node leaves the edge cloud
Latitude, Longitude	The longitude and latitude of the base station where the mobile phones access the Internet
User ID	Volunteer edge node ID

V. COMPUTATIONAL STUDY

In this section, we evaluate the proposed approach based on Shanghai Telecom's base station data set[2]. First, we present the dataset. Second, we validate the proposed XGBoost, compared with other regression models such as Random Forest and ANN. Third, we discuss the trust-aware scheduling performance compared to the random approach.

A. DATASET

The Shanghai Telecom's base station dataset encompasses 6,236,620 samples from 9,481 mobile devices connected to 3,233 base stations as illustrated in Figure 5. It covers a the period from May to December 2014, and provides details such as devices' locations and internet user records. Therefore, it is adopted in numerous studies on resource management within edge and mobile edge computing [38], [39], [40], [41]. In our study, we interpret the users in this dataset as volunteer edge nodes. Thereby, we obtain the parameters start time, end time, and location in (see Table 3).

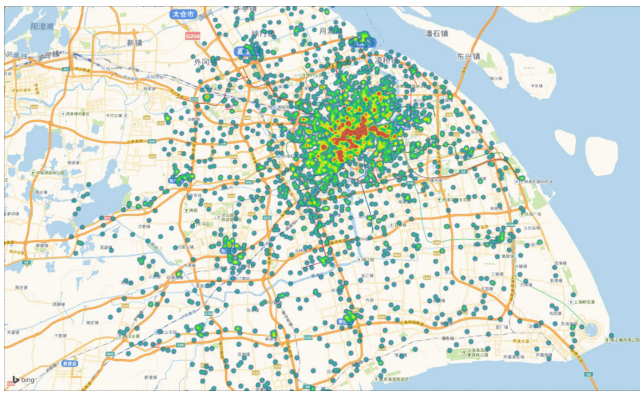

FIGURE 5. Dataset.

B. XGBOOST VALIDATION

We validate both the lag-based and windows-based approach of the proposed XGBoost. Different statistical metrics were used, such as, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R squared (R2). The performance comparison between the proposed XGBoost, Random Forest, and Artificial Neural Network (ANN) with lag-based features is shown in Table 4. The RMSE value of XGBoot is 6.32, smaller than random forests, and the ANN models' RMSE are 13.64 and 18.3. Similarly to MAP, and R2, XGBoost shows higher accuracy than other models. For all metrics, RMSE, MAP, and R2 XGBoost with lag-based show better and more accurate results

[2]http://sguangwang.com/TelecomDataset.html

TABLE 4. Performance comparison between the proposed XGBoost, Random Forest, and ANN with lags-based features.

Model	RMSE	MAP	R2
XGBoost	**6.32**	**3.9**	**0.98**
Random Forest	13.64	7.73	0.93
ANN	18.3	10.35	0.88

TABLE 5. Performance comparison between the proposed XGBoost, Random Forest and ANN with window-based features.

Model	RMSE	MAP	R2
XGBoost	**51.16**	**38.368**	**0.10**
Random Forest	53.78	40.73	0.03
ANN	58.70	33.66	-0.15

The performance comparison between the proposed XGBoost, Random Forest, and ANN with window-based features is shown in Table 5. For all metrics, RMSE, MAP, and R2 XGBoost show significantly lower prediction errors and better correlation. However, XGBoost with lag-based approach outcomes outperforms the window-based approach because the lag-based rely on availability differences in relation to prior behavior.

The predicted availability value along with the actual availability values for some edge nodes are illustrated in Figure 6. As seen in figure, the predicted availability value follows the availability values well enough because of the sufficient RMSE and R2 values.

C. SIMULATION

Since building, managing, and maintaining a network of edge servers is costly and time-consuming, we use a discrete-event simulation to mimic the dynamics of the system and thereby evaluate and compare the performance of our overall decision-making procedure. The simulation was built in Python with the help of the SimPy library [42].

1) EXPERIMENTAL SETUP

The simulation takes place on a map that has multiple regions (e.g. 11 edge regions) where the edge nodes and tasks spawn on the map. The regions are defined as a grid where the

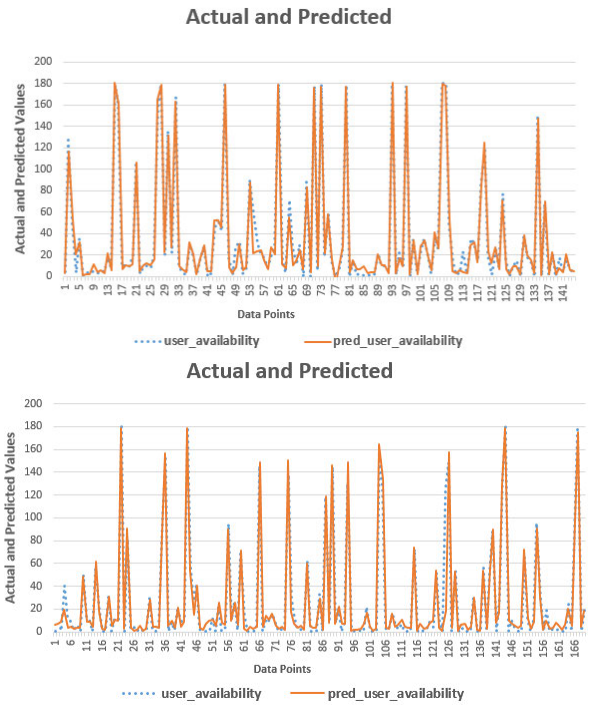

FIGURE 6. The actual and predicted availability value for different volunteer edge nodes.

number of rows and columns can be specified. The edge node distributions are derived from the dataset. The arrivals of node in each region follow a Poisson process. Hence, the inter-arrival times are exponentially distributed. We vary the arrival rate of volunteers between 0.3 and 8.5 minutes across regions to generate regions with different volunteer densities. The arrivals of tasks is modelled with a Poisson process. Hence, the inter-arrival-times are exponentially distributed. The duration until the deadline is assumed to follow a triangular distribution that is identical in all regions, where the minimum duration is 100 seconds, the maximum duration is 400 seconds, and the mode of the distribution is 250 seconds. The processing duration of tasks is assumed to follow a tiangular distribution which is identical in all regions, where the minimum duration is 70 seconds, the maximum duration is 200 seconds, and the mode of the distribution is 150 seconds. The tasks are assigned to the edge nodes according to the proposed assignment logic. We can only assign a task to an edge node when the task is within the node's reach. Task and resource events and states is discussed above. We generate task dependencies by randomly selecting a number of requirements from the environment's existing tasks. The number of dependencies is drawn from a Poisson distribution with a mean of one and is limited to a maximum of two in our experiment. During the simulation, we observe the number of failed dependent tasks, total task failures, task completions, total latency time, and average execution time. These are the most critical performance factors to consider while designing a computer network. Each experiments is conducted over 21200 seconds.

2) COMPARED APPROACHES

We compare our approach, which we refer to as "Trust-aware approach", to several alternative algorithms, including the Beta-based trust approach [6], FCFS Approach, Random Approach, and Round-Robin Approach. Moreover, as our approach considers the likelihood that other volunteer nodes close to the selected one are available for successor tasks, we analyze effect of explicitly considering this aspect in our metric. To that end, compare our original approach to a variant where the 2nd term of our metric is omitted. We call this variant "Trust-aware without nearby trusts". Now we briefly discuss how to use the existing approaches to assign tasks to volunteers.

- **Beta-based trust approach:** Beta consists of two main parameters that shape the distribution: α, and β, representing the number of completed and failed tasks, respectively. The task is assigned to the most trusted edge node.
- **First come first server (FCFS)**: The task is assigned to first available volunteer edge node.
- **Random Approach**: The task is assigned to random volunteer edge node.
- **Round-Robin Approach**: The tasks are submitted to node based on a cyclic rotation. The approach selects the node that has had the least recent assignment, ensuring a fair distribution of tasks among available idle nodes.

3) THE NUMBER OF FAILED DEPENDENT TASKS

In this section, we present an in-depth comparison between the proposed "Trust-aware approach" and the "Trust-aware without nearby trusts" methodology. Figure 7 presents an overview of the total number of failed dependent tasks across different edge regions. A clear pattern can be observed. Specifically, our proposed method consistently outperforms the alternative approach, which overlooks neighborhood trusts. The superiority of our method is evident in the significantly lower proportion of failed dependent tasks. We observe percentage variations in the number of failed dependent tasks ranging from 25.33% and 60.82% between the two approaches. On average, our proposed method reduces the number of unsuccessful dependent tasks by

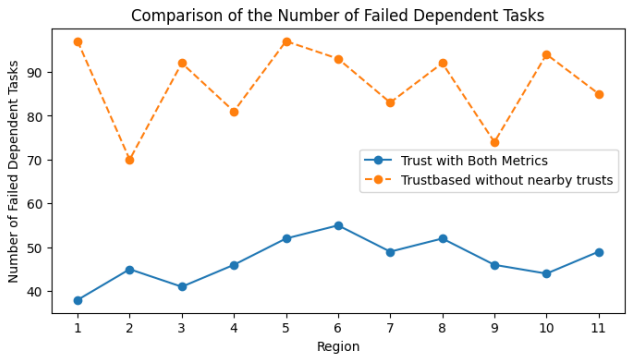

FIGURE 7. Comparison of the number of failed dependent tasks.

approximately 43.44%. These results highlight the effectiveness of our strategy in managing task dependencies and mitigating failures. In Figure 7, regions are arranged from high to low density, where high-density regions represent areas with a higher number of available resources. Region 1 serves as a typical example of high-density areas. The enhanced performance observed in high-density regions, like Region 1, stems from the greater availability of nearby volunteers. Conversely, when such areas don't effectively utilize these volunteers, as demonstrated in the 'Trust-aware without nearby trusts' scenario, there is a pronounced risk of task failures.

4) TOTAL TASK FAILURES AND TASK COMPLETIONS

This section presents a comprehensive comparison between our technique and several other algorithms, including the Beta technique, FCFS Approach, Random Approach, and Round-Robin Approach. To evaluate the effectiveness of our strategy, in this section we focus on two crucial metrics: The total number of successfully completed tasks and the total number of failed tasks across each region.

In Figure 8 Our approach consistently outperforms all other algorithms in terms of task completion across all regions. For instance, when compared to the Random Approach, we see an average improvement of 3.54%. Furthermore, we achieve a 7% improvement over the Beta approach. As depicted in Figure 8, task completions tend to decrease while moving from high-density regions to low-density areas for all algorithms. This trend arises because high-density regions have a greater abundance of resources, leading to a higher likelihood of task assignment and completion.

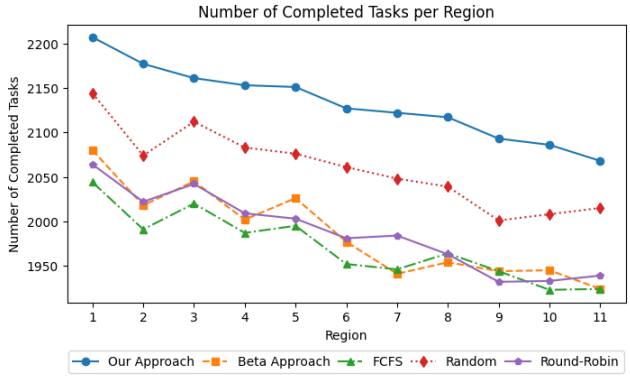

FIGURE 8. Comparison of number of completed tasks per region.

Furthermore, In Figure 9 our approach delivers exceptional outcomes in terms of task failures, with an average of only 19 failures. Compared to the Beta Approach (average of 166 tasks), the FCFS Approach (average of 187 tasks), the Random Approach (average of 97 tasks), and the Round-Robin Approach (average of 174 tasks), this represents a remarkable reduction of 87.7%, 88.7%, 78.4%, and 87.9%. These findings emphasize the effectiveness with

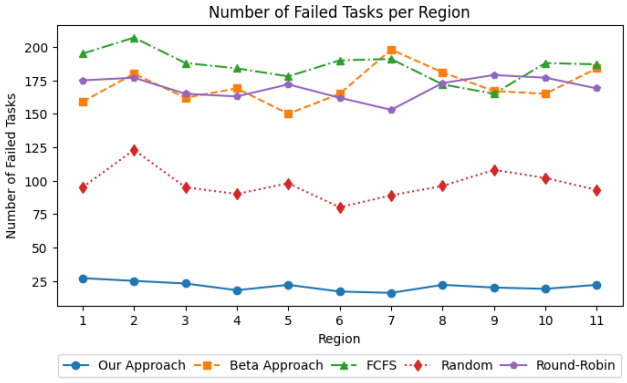

FIGURE 9. Comparison of number of failed tasks per region.

which our approach can minimize task failures and significantly decrease the number of unsuccessful tasks.

5) TOTAL LATENCY TIME AND AVERAGE EXECUTION TIME

We discuss the results of total delay time and average execution time. Our approach outperforms all other algorithms in terms of total delay time. Our approach achieves an average total delay time of 1402.33 seconds across all edge regions, compared to 2769.46 seconds for the Beta approach. The Random approach records 2928.25 seconds, while the Round-Robin approach achieves 2761.51 seconds and the FCFS approach shows a worse performance with 2986.23 seconds. These results demonstrate the effectiveness of our approach in minimizing task delays, resulting in significantly shorter delay durations of approximately 47.32%.

In Figure 10, region 1 and 2 shows the highest latency with our method, However, Figures 8 and 9 depict that our approach results in superior task completion and diminished failure rates. The longer latency in region 1 and 2 results from our strategy of offloading tasks to nearby trusted resources after initial failures. While this adds latency, it significantly enhances task completion rates and decreases failures. In our analysis region 9 notably exhibited lower delays compared to other algorithms. Region 9, in particular, benefits from the efficient utilization of resources, even in areas of low resource density. This efficient use ensures that tasks are promptly addressed, minimizing delays further.

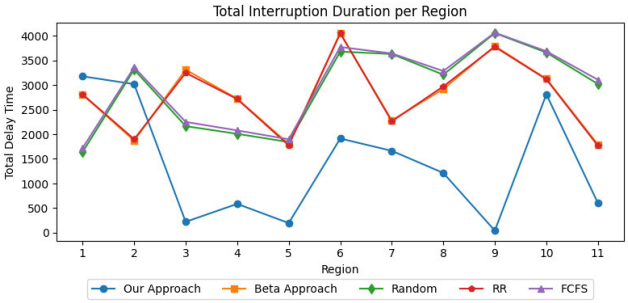

FIGURE 10. Comparison of total delay time per region.

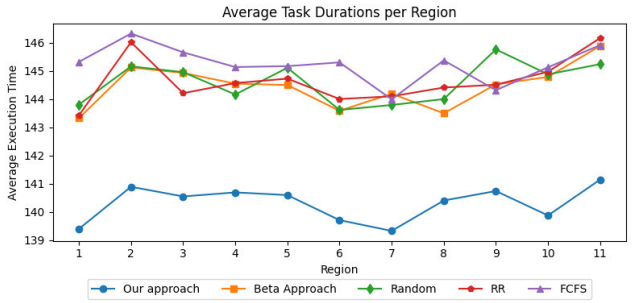

FIGURE 11. Comparison of average execution time per region.

Furthermore, in terms of average execution time, our approach is found to the alternative tested algorithms. The average execution time in our approach indicating a reduction of around 2.87% compared to the Beta Approach, 2.96% compared to the Random Approach, 3.00% compared to the Round-Robin Approach, and 3.40% compared to the FCFS Approach. These results reveal that our approach outperforms other algorithms in terms of both total delay time and average execution time. As such, we can conclude that it is efficient and effective in task management and execution. This indicates that our approach is more efficient in completing tasks within a shorter time frame by both likelihood of individual and nearby that play a significant positive impact on minimizing the delay. While the average execution time is currently calculated without considering the delay, if we were to include the delay in our calculations, the resulting number would be significantly higher

Based on a thorough analysis of the results, it is evident that our approach consistently surpasses other algorithms in various performance metrics, including completed tasks, task failure reduction, delay minimization, and average execution time. The capability of our model to balance the trust score between the volunteer candidate and its neighbors shows significant improvement over other algorithms that rely solely on the direct trust of the candidate. Furthermore, our approach takes into account tasks with dependencies and assigns them to volunteers equipped to handle both the task and its dependencies. Trust calculations based on availability ensure that tasks are not assigned to resources that might terminate soon, while tasks nearing their deadlines are given priority.

VI. LIMITATION AND FUTURE WORK

In this subsection, we briefly discuss some limitations of our approach. Our approach primarily focuses on evaluating trust-aware scheduling, the behavior, and availability of volunteers as resources for task execution. Thus, the model assumes that tasks are submitted to volunteer edge nodes rather than local devices or cloud data centers. Future research directions include addressing challenges related to scarce resources such as energy and communication infrastructure. Expanding the framework to include volunteer capacities, resource utilization, and power consumption would further enhance its capabilities.

VII. CONCLUSION

This paper presented a novel machine learning-based approach for trust-aware scheduling in volunteer edge computing. By incorporating a metric that measures trust based on the likelihood of individual volunteers completing tasks and the availability of their neighbors, we extended the notion of trust in volunteer edge computing. Our approach outperformed baseline algorithms in terms of completed tasks and minimized task failures, and delay. The experiments conducted using Telecom's base station dataset have provided compelling evidence regarding the effectiveness of our approach. Our approach consistently outperformed other algorithms, resulting in significant improvements across various metrics. Specifically, our approach achieved an average reduction of 43.44% in the number of failed dependent tasks, an increase of 5% in completed tasks, a decrease of 85.68% in failed tasks, a reduction of 47.32%. in delay duration, and a decrease of 3.06% in average execution time compared to alternative algorithms. These results clearly highlight the superior performance of our approach in trust-aware scheduling and resource management within volunteer edge computing environments. The presented machine learning model, used to predict the variables for computing trust, played a crucial role in the performance of our approach. By leveraging this predictive capability, we made informed decisions in real-time during task assignments, resulting in efficient and intelligent resource allocation.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Scientific Research (DSR) at King Abdulaziz University for their technical and financial support.

REFERENCES

[1] I. Al Ridhawi, M. Aloqaily, and Y. Jararweh, "An incentive-based mechanism for volunteer computing using blockchain," *ACM Trans. Internet Technol.*, vol. 21, no. 4, pp. 1–22, Nov. 2021.

[2] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system," *ACM Trans. Internet Technol.*, vol. 21, no. 4, pp. 1–21, Nov. 2021.

[3] M. I. Bhat and K. J. Giri, "Impact of computational power on cryptography," in *Multimedia Security*. Cham, Switzerland: Springer, 2021, pp. 45–88.

[4] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3247–3257, Jun. 2021.

[5] T. M. Mengistu, A. Albuali, A. Alahmadi, and D. Che, "Volunteer cloud as an edge computing enabler," in *Proc. Int. Conf. Edge Comput.* Cham, Switzerland: Springer, 2019, pp. 76–84.

[6] Y. S. Alsenani, G. V. Crosby, K. R. Ahmed, and T. Velasco, "ProTrust: A probabilistic trust framework for volunteer cloud computing," *IEEE Access*, vol. 8, pp. 135059–135074, 2020.

[7] Y. Alsenani, G. V. Crosby, K. R. Ahmed, and T. Velasco, "Towards multi-criteria volunteer cloud service selection," in *Proc. Int. Conf. Cloud Comput.* Honolulu, HI, USA: Springer, Sep. 2020, pp. 278–286.

[8] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–37, Sep. 2020.

[9] Y. Alsenani, G. Crosby, and T. Velasco, "SaRa: A stochastic model to estimate reliability of edge resources in volunteer cloud," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 121–124.

[10] L. Fotia, F. Delicato, and G. Fortino, "Trust in edge-based Internet of Things architectures: State of the art and research challenges," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–34, Sep. 2023.
[11] P. Srikanth, A. Kumar, and M. Hedabou, "An uncertainty trust assessment scheme for trustworthy partner selection in online games," *IEEE Access*, vol. 10, pp. 132232–132249, 2022.
[12] K. A. Awan, I. U. Din, A. Almogren, and J. J. P. C. Rodrigues, "AutoTrust: A privacy-enhanced trust-based intrusion detection approach for Internet of smart things," *Future Gener. Comput. Syst.*, vol. 137, pp. 288–301, Dec. 2022.
[13] M. L. Alarcon, M. Nguyen, A. Pandey, S. Debroyand, and P. Calyam, "VECFlex: Reconfigurability and scalability for trustworthy volunteer edge-cloud supporting data-intensive scientific computing," in *Proc. IEEE/ACM 15th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2022, pp. 151–156.
[14] T. M. Mengistu and D. Che, "Survey and taxonomy of volunteer computing," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–35, May 2020.
[15] Y. Alsenani, G. V. Crosby, T. Velasco, and A. Alahmadi, "ReMot reputation and resource-based model to estimate the reliability of the host machines in volunteer cloud environment," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2018, pp. 63–70.
[16] M. Alazab, R. M. S. Priya, M. Parimala, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: Concepts, challenges, and future directions," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3501–3509, May 2022.
[17] O. A. Wahab, G. Rjoub, J. Bentahar, and R. Cohen, "Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems," *Inf. Sci.*, vol. 601, pp. 189–206, Jul. 2022.
[18] G. Rjoub, O. A. Wahab, J. Bentahar, and A. Bataineh, "Trust-driven reinforcement selection strategy for federated learning on IoT devices," *Computing*, pp. 1–23, Apr. 2022.
[19] L. Zhang, Y. Zou, W. Wang, Z. Jin, Y. Su, and H. Chen, "Resource allocation and trust computing for blockchain-enabled edge computing system," *Comput. Secur.*, vol. 105, Jun. 2021, Art. no. 102249.
[20] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.
[21] M. H. ur Rehman, K. Salah, E. Damiani, and D. Svetinovic, "Towards blockchain-based reputation-aware federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 183–188.
[22] X. Cao, M. Fang, J. Liu, and N. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. NDSS*, 2021, pp. 1–18.
[23] H. Peng, W.-S. Wen, M.-L. Tseng, and L.-L. Li, "Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment," *Appl. Soft Comput.*, vol. 80, pp. 534–545, Jul. 2019.
[24] F. Yucel, M. Yuksel, and E. Bulut, "QoS-based budget constrained stable task assignment in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 11, pp. 3194–3210, Nov. 2021.
[25] R. Latif, M. U. Ahmed, S. Tahir, S. Latif, W. Iqbal, and A. Ahmad, "A novel trust management model for edge computing," *Complex Intell. Syst.*, vol. 8, pp. 3747–3763, Oct. 2021.
[26] R. Agliamzanov, M. Sit, and I. Demir, "Hydrology@home: A distributed volunteer computing framework for hydrological research and applications," *J. Hydroinformatics*, vol. 22, no. 2, pp. 235–248, Mar. 2020.
[27] A. Souri, Y. Zhao, M. Gao, A. Mohammadian, J. Shen, and E. Al-Masri, "A trust-aware and authentication-based collaborative method for resource management of cloud-edge computing in social Internet of Things," *IEEE Trans. Computat. Social Syst.*, early access, Feb. 7, 2023, doi: 10.1109/TCSS.2023.3241020.
[28] A. Waheed, M. A. Shah, A. Khan, and G. Jeon, "An infrastructure-assisted job scheduling and task coordination in volunteer computing-based VANET," *Complex Intell. Syst.*, vol. 9, pp. 3613–3633, Jun. 2021.
[29] C. Bayliss, J. Panadero, L. Calvet, and J. M. Marquès, "Reliability in volunteer computing micro-blogging services," *Future Gener. Comput. Syst.*, vol. 115, pp. 857–871, Feb. 2021.
[30] S. Gonzalo, J. M. Marqués, A. García-Villoria, J. Panadero, and L. Calvet, "CLARA: A novel clustering-based resource-allocation mechanism for exploiting low-availability complementarities of voluntarily contributed nodes," *Future Gener. Comput. Syst.*, vol. 128, pp. 248–264, Mar. 2022.
[31] B. Ali, M. A. Pasha, S. u. Islam, H. Song, and R. Buyya, "A volunteer-supported fog computing environment for delay-sensitive IoT applications," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3822–3830, Mar. 2021.
[32] E. Alemneh, S.-M. Senouci, P. Brunet, and T. Tegegne, "A two-way trust management system for fog computing," *Future Gener. Comput. Syst.*, vol. 106, pp. 206–220, May 2020.
[33] A. Josang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Electron. Commerce Conf.*, vol. 5, 2002, pp. 2502–2511.
[34] A. Celestini, A. L. Lafuente, P. Mayer, S. Sebastio, and F. Tiezzi, "Reputation-based cooperation in the clouds," in *Proc. IFIP Int. Conf. Trust Manag.* Singapore: Springer, Jul. 2014, pp. 213–220.
[35] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov, "Decentralized self-enforcing trust management system for social Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2690–2703, Apr. 2020.
[36] J. I.-Z. Chen, "Embedding the MRC and SC schemes into trust management algorithm applied to IoT security protection," *Wireless Pers. Commun.*, vol. 99, no. 1, pp. 461–477, Mar. 2018.
[37] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[38] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.
[39] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Software, Pract. Exper.*, vol. 50, no. 5, pp. 489–502, May 2020.
[40] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 55–67, Jan. 2022.
[41] Z. Li, G. Li, M. Bilal, D. Liu, T. Huang, and X. Xu, "Blockchain-assisted server placement with elitist preserved genetic algorithm in edge computing," *IEEE Internet Things J.*, early access, Jun. 28, 2023, doi: 10.1109/JIOT.2023.3290568.
[42] *SimPy*. Accessed: Sep. 5, 2022. [Online]. Available: https://simpy.readthedocs.io/en/latest/

YOUSEF ALSENANI received the Ph.D. degree in computer science from Southern Illinois University Carbondale, in 2021. He is currently an Assistant Professor of information systems with King Abdulaziz University (KAU), Saudi Arabia, and the Research and Development Director and the Head of the AI and Privacy, Lun Startup Studio. He has successfully developed numerous AI and privacy algorithms that have been tested within highly sensitive entities in Saudi Arabia. He has published several peer-reviewed publications in prestigious IEEE conferences and journals and worked as a PI and a Co-PI on various national research projects. His research interests include cloud and edge computing, trust, security, and federated learning. He is a reviewer in top computer science journals, including IEEE TRANSACTIONS.

ABDULAZIZ S. ALNORI received the Ph.D. degree from the University of Leeds, U.K. He is currently an Assistant Professor with the Faculty of Computing and Information Technology, King Abdulaziz University. His research interests include adopting machine learning and deep learning in solving problems, quality of service, trust management, and the Internet of Things.