IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 18, 2019, accepted October 1, 2019, date of publication October 14, 2019, date of current version October 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947306

A Trust-Aware Task Offloading Framework in Mobile Edge Computing

DEXIANG WU[1], GUOHUA SHEN[1,2,3], ZHIQIU HUANG[1,2,3], YAN CAO[1], AND TIANBAO DU[1]

[1]College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210093, China
[3]Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology, Nanjing 211106, China

Corresponding author: Guohua Shen (ghshen@nuaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772270, in part by the National Key Research and Development Program of China under Grant 2018YFB1003902 and Grant 2016YFB1000802, and in part by the Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology of China under Grant 1015-XCA1816403.

ABSTRACT Task offloading in Mobile Edge Computing (MEC) is a solution to augment resource-limited mobile devices' capabilities by migrating tasks to the edge of the network (i.e., edge servers and idle devices). At present, a lot of work is focused on optimizing policies to reduce latency or energy consumption for users. However, they mostly ignore that services are not necessarily trustworthy because the resource providers are complex, dynamic, and unreliable. The trustworthiness of a service in our paper mainly includes two aspects. One is that resource providers will not violate users' privacy. The other is that resource providers will perform well to ensure the effectiveness of services. To solve this problem, we propose a trust-aware task offloading framework. The main purpose of the framework is to select a resource provider for a user to reduce latency or energy consumption and ensure service trustworthiness at the same time. The framework can be divided into three modules (i.e., trust evaluation, filtering and selection). By combining trust evaluation and filtering modules, some resource providers that are not trusted by users are filtered out to ensure that the services provided to users are trustworthy. In the selection module, we select an appropriate provider for a user from the qualified (i.e., left after the filtering process) resource providers based on an offloading policy. The experimental results show that our framework not only reduces latency or energy consumption for users, but also reduces the failure rate of tasks.

INDEX TERMS Mobile edge computing (MEC), task offloading, trust evaluation, machine learning, privacy protection.

I. INTRODUCTION

In the past decade, cloud computing has developed rapidly and brought many benefits. Computation offloading in cloud computing allows users to offload computationally intensive tasks to a resource-rich remote cloud [1]. With the rapid growth of smart mobile devices such as smartphones and wearable devices, more and more intelligent mobile applications such as face recognition, virtual reality, health surveillance, and real-time translation are emerging [2]. These applications usually run resource-hungry algorithms (e.g., GPU rendering and deep learning) [3], which would require intensive computing and high energy consumption. In addition, these emerging applications often demand low-latency. However, cloud computing cannot meet the above two requirements at the same time [4].

To address this challenge, Mobile Edge Computing (MEC) as a complement to cloud computing has been proposed to solve the shortcomings of cloud computing. By sinking computing power to the edge of the network and leveraging a multitude of end-user devices to complete computation tasks [5], it promises to provide users with low-latency services.

Task offloading in MEC is a solution to augment resource-limited mobile devices' capabilities by migrating tasks to the edge of the network. D2D (device-to-device) communication has become a key technology of 5G, and it is widely added to mobile edge computing. At present, a lot of research has been carried out on task offloading in MEC, and good results have been achieved in reducing latency and energy consumption. However, only a little work has focused on whether

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

the services provided are trustworthy. Offloading in MEC is more complex than that in cloud computing. The resource providers in cloud computing are usually several large companies, while the resource providers in MEC can be either MEC servers or idle devices around the network. Therefore, resource providers in MEC have new characteristics such as dynamic, complex, and unreliable, which make it difficult to ensure that services are trustworthy.

The trustworthiness of a service in our paper mainly includes two aspects. One aspect is that resource providers will not violate users' privacy, which we refer to as *Priv − trustworthiness*. The essence of task offloading is the outsourcing of computation, so the privacy of users must also be protected in the process. When a task contains sensitive information, it will pose a threat to personal privacy if such information is acquired by malicious devices and used maliciously. Zhang *et al.* [6]–[8] pointed out that since user data in mobile edge computing is usually processed in semi-trusted authorized entities, it is likely to cause data leakage or data loss. The other is that resource providers will perform well to ensure the effectiveness of services, which we refer to as *Effe − trustworthiness*. If the number of misbehaving providers increases, there will be some bad situations, such as high task failure rate or actual latency far beyond prediction, which will disrupt offloading and lead to low offloading effectiveness. For example, a resource provider who only wants to benefit from task offloading does not contribute resources or reduces a large number of resources without agreement. If we do not consider the trustworthiness of services, even if offloading policies achieve good results in the optimization of latency or energy consumption, it is difficult to apply task offloading in practice because of the low effectiveness or the potential threat to users' privacy.

To ensure the trustworthiness of services (i.e., *Priv − trustworthiness* and *Effe − trustworthiness*), we conduct a trust evaluation of these two aspects. On the one hand, to protect a user's privacy, we introduce task sensitivity to measure the threat to the user's privacy caused by the exposure of sensitive information contained in the task, and introduce identity trust to evaluate the trust degree of the user to providers. When the task sensitivity is higher than the identity trust level of a provider, it indicates that the provider is not trusted by the user to complete the task. On the other hand, to ensure the effectiveness of services, we introduce behavior trust to evaluate whether providers are misbehaving. Users have their required behavior levels of services. When a provider's behavior trust level is lower than a user's required behavior level, it is not trusted by the user to complete the task. However, it is difficult to choose an appropriate weight for each trust factor in establishing a trust evaluation mechanism. Therefore, we establish a novel trust evaluation mechanism by formalizing trust evaluation as a classification problem in machine learning.

Overall, to reduce latency or energy consumption and ensure the trustworthiness of services at the same time, we propose a trust-aware task offloading framework. The framework can be divided into three modules (i.e., trust evaluation, filtering, and selection). In the trust evaluation module, we introduce identity trust and behavior trust to evaluate resource providers, and establish a novel trust evaluation mechanism based on machine learning. In the filtering module, we filter out some resource providers with low trust level by filtering algorithm based on the results of trust evaluation. Based on the two modules, we are able to ensure the trustworthiness of services. In the selection module, we select the appropriate one from the qualified resource providers for the user to perform the task according to the offloading policy.

The main contributions of the paper are as follows: (1)

1) To ensure the *Priv − trustworthiness*, we introduce task sensitivity to measure the threat to the user's privacy if the sensitive information contained in the task is exposed. We introduce online social networks into MEC, which maps the trust relationship established by device owners in the social layer into the identity trust relationship between devices. If a provider's identity trust level is lower than task sensitivity, it is untrustworthy. To ensure the *Effe − trustworthiness*, we introduce behavior trust to evaluate the behavior performance of resource providers. Users have the required behavior levels of services. If a provider's behavior trust level is lower than the user's required behavior level, it is untrustworthy.
2) Based on the relationship between task sensitivity and identity trust, as well as the relationship between behavior level required by the user and behavior trust level of a provider, a trust-based filtering algorithm is proposed to filter out those unqualified providers.
3) We propose a trust-aware task offloading framework to reduce latency or energy consumption and ensure the trustworthiness of services for users at the same time.

The remainder of this paper is organized as follows: Section II briefly introduces the related work. Section III describes the relevant preliminary knowledge. Section IV introduces our trust-aware task offloading framework in detail. Section V introduces the experimental evaluation of our framework. Finally, we make a summary and put forward the direction of future research.

II. RELATED WORK

To select trustworthy resource providers, we need to establish a trust evaluation mechanism. Meanwhile, facing a large number of resource providers, we need to select an appropriate resource provider to complete a task according to an offloading policy. Therefore, in this section, we focus on trust evaluation and task offloading policies in MEC.

A. TRUST EVALUATION IN MOBILE EDGE COMPUTING

Trust evaluation has received extensive attention and research in areas including cloud computing, mobile cloud computing, and peer-to-peer computing [9]–[12]. At present, people are building some new trust evaluation mechanisms based on the new characteristics of MEC.

According to the sources of trust factors, we classify the existing evaluation work in MEC into three categories: indirect trust models (based on reputation), direct trust models (based on direct interaction records), and hybrid models (both direct and indirect trust factors). Hussain and Almourad [13], [14] studied how to compute the reputation of edge data centers, using centralized trust management, which stores the reputation of LTE deployed clouds. Through this system, users can anonymously evaluate cloudlet services. Zhou *et al.* [15] proposed a method to learn users' preferences based on their contexts, previous behaviors, and social intimacy. He designed a trust evaluation mechanism based on users' preferences to guarantee trustworthy edge computing. To enhance the reliability of trust evaluation, we will take into account both indirect trust factors and dircet trust factors.

According to the technology used, we divide the existing evaluation work in MEC into two categories: emerging technology-based models and formula-based models. Xu *et al.* [16] presented a novel trustless crowd-intelligence ecosystem based on the common decentralization feature of mobile edge computing and blockchain technology, which can ensure that all trustless objects in the whole system must perform trusted operations. The formula-based models always assign a weight to each trust factor and integrate them into a value. Yuan and Li [17] made some discussions on trust management in mobile edge computing, and elaborated the reasons and significance of trust management. He proposed a trust computing mechanism based on multi-source feedback (*MSTrust*).

However, in the traditional formula method, how to give each trust factor an appropriate weight is a very difficult problem. Although the value derived from the formula-based approach can be used as a reference for the trust level, it is not accurate enough to be in accordance with the user's expectations because trust is a subjective metric. Besides, they are not easily understood by users directly. It is necessary to provide users with an accurate and intuitive trust evaluation method. Compared with the above solutions, trust evaluation in our framework is formalized as a classification problem in machine learning.

Based on the above analysis, we will take hybrid trust factors into consideration and utilize a classification approach in machine learning to establish a trust evaluation mechanism for mobile edge computing.

B. TASK OFFLOADING POLICIES IN MOBILE EDGE COMPUTING

So far, task offloading has gained a lot of researchers' attention. An offloading policy is an important part of offloading process, determining which tasks will be offloaded and where they will be offloaded.

From the existing work, the objectives of offloading policies are mainly divided into three types: reducing latency, reducing energy consumption, and balancing latency and energy consumption [18]. Zhang *et al.* [19] proposed an optimal offloading scheme with the goal of reducing latency in the case of limited computing resources of MEC. The author proposed a layered MEC deployment architecture and solved the multi-user offloading problem by using Stackelberg game theory. The author's proposed offloading scheme not only provides a significant improvement in reducing latency over local execution, but also guides the deployment of MEC servers. Liu *et al.* [20] introduced a cloud-assisted edge computing framework with a three-tier network, and the energy consumption minimization problem is formalized as a mixed integer programming. Dong [21] proposed a dynamic, decentralized resource allocation policy based on latency and energy consumption, which is used to deal with the problem of task offloading between multiple heterogeneous edge nodes and central clouds.

In addition to the above-mentioned offloading policies to reduce latency or energy consumption from the perspective of selecting the appropriate resource provider, there are other interesting research on optimizing offloading policies from the perspective of communication and computation cooperation. Li *et al.* [22] exploited the idea of computation replication which allows each user to offload its task to multiple edge nodes to speed up the downloading phase via transmission cooperation. Their results show that computation replication is very useful for reducing communication latency in tasks where the output data size is larger than the input data size. On the contrary, sometimes multiple users can use collaborative properties to share part of a user's computation tasks. Al-Shuwaili and Simeone [23] leveraged the inherent collaborative properties of Augmented Reality (AR) applications and proposed a novel resource allocation approach over both communication and computation resources. In this way, different users are able to share part of the computation tasks as well as the input and output data. Our research is to optimize the policy by selecting an appropriate resource provider for the user to complete the task.

Based on the above analysis, the offloading policy in our framework aims to select the appropriate resource provider for the user, taking into account latency and energy consumption.

III. PRELIMINARIES

Corresponding to the two aspects of service trustworthiness, we need to introduce task sensitivity, identity trust evaluation, and behavior trust evaluation.

Task sensitivity and identity trust evaluation are introduced to ensure the $Priv - trustworthiness$. We compare the task sensitivity level with the identity trust level of a provider to protect the privacy of the user. If the task sensitivity is higher than the identity trust level of a provider, it indicates that the provider is unqualified to complete the task.

Behavior trust evaluation is introduced to ensure the $Effe - trustworthiness$. We compare the behavior level required by the user and the behavior trust level of a provider to ensure the effectiveness of services. If the behavior trust level of a provider is lower than the user's required behavior level, the provider is unqualified. These providers who are

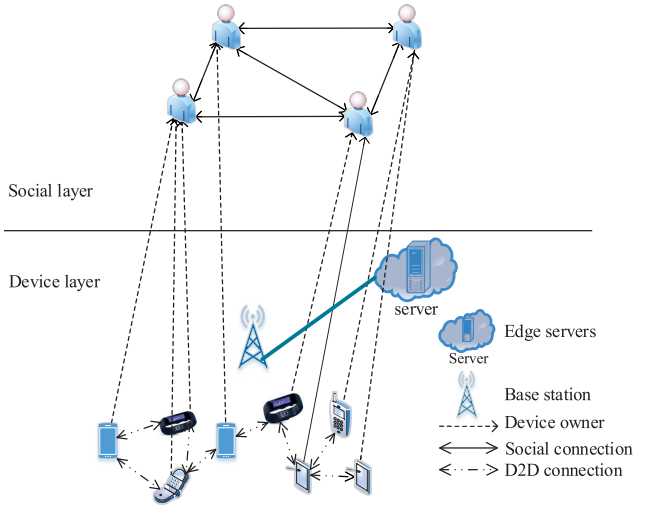


FIGURE 1. Social collaborative mobile edge computing.

not qualified in identity or behavior will be removed. In this section, we will explain the above concepts, which are prerequisites closely related to our framework.

A. DEFINITION OF SOCIAL COLLABORATIVE MOIBLE EDGE COMPUTING

Online social networks have become an indispensable part of people's daily life and have been widely used in many scenarios, such as friend recommendation and access control system based on trust.

To protect the privacy of users in task offloading, we introduce online social networks into MEC. We call this new paradigm, which combines mobile edge computing with online social networks, Social Collaborative Mobile Edge Computing (SCMEC). Because mobile devices, wearable devices, and so on are owned and carried by people, it is hopeful and promising to construct a secure collaborative environment by leveraging social relationships formed by people's social interactions. For one thing, it is able to increase people's willingness to contribute their resources; for another, based on social relationships, selecting trusted nodes for task offloading enhances privacy protection for users.

As shown in Fig. 1, the diagram is divided into a device layer and a social layer. In the social layer, device owners establish trust relationships with other people through social applications and have different trust levels for different people. Everyone can own one or more devices. In the device layer, the trust relationships between devices correspond to the trust relationships between their owners in the social layer. A device can establish a D2D connection with another device within a certain range. Devices can request services from nearby MEC servers or nearby devices to complete tasks. It should be noted that, if there is no special explanation, the MEC mentioned below refers to SCMEC.

B. RELATED THEORIES IN SOCIAL COLLABORATIVE MOIBLE EDGE COMPUTING

1) TASK SENSITIVITY

Our paper refers to the Health Insurance Portability and Accountability Act (HIPAA) [24] standard to determine which personal information is sensitive. Here are some related concepts about task sensitivity.

- Sensitivity of personal information: It is the extent to which users are concerned about providing private data. It measures the damage to users when sensitive information items are exposed. The higher the sensitivity of personal information, the greater the harm to users when it is exposed. SI is a finite set of personal information items. For any personal information item $s \in SI$, the function $f(s)$ obtains its sensitivity. We divided the sensitivity of personal information into three levels. In other words, $f(s) \in \{1, 2, 3\}$. The values from 1 to 3 represent low, medium, and high, respectively. If s is level 3, it means that when s is exposed, it will pose a threat to the user's property or life. If s is level 2, it means that when s is exposed, it will cause discomfort to the user. If s is level 1, it means that when s is exposed, it will have little impact on the user. For more detailed information, Weible [25] analyzed which personal information is more sensitive to users. Sensitivity is suggested by relevant experts and can be changed according to the user's own requirements.
- Sensitive information set: It refers to a set of several sensitive information items.
- Relevant sensitivity: It refers to the harm degree to users when two personal information items s_i and s_j are exposed simultaneously. It can be obtained by the function $F(s_i, s_j)$. $F(s_i, s_j) \in \{1, 2, 3\}$. Values ranging from 1 to 3 represent low, medium, and high, respectively. Relevant sensitivity is given by experts after reasoning and analysis, and is not able to be modified by users.
- Task sensitivity: A task T_t consists of a piece of code and some input data to perform a specific function. There are some personal information items in the input data. The personal information items that need to be collected for a task to perform a specific function are predetermined and limited in number. We determine the sensitivity of a task by analyzing the sensitivity of these information items. The sensitivity of a task can be calculated by using (1), and n is the number of personal information items contained in the task. $1 \leq i \leq n$ and $i + 1 \leq j \leq n$. When a task contains only one sensitive information item, task sensitivity is the sensitivity level of the information item. When a task contains multiple sensitive information items, it is necessary to consider the relevant sensitivity level of each two items and the respective sensitivity levels of the two items. Task sensitivity is the maximum of these three levels in all combinations. In particular, when a task does not involve any personal sensitive information, the sensitivity of the

task is 0, that is, the task is not sensitive. If there are n personal information items needed in a task, the number of relevant sensitivity composed of each two personal information items is C_n^2. Therefore, the computation of task sensitivity has a low complexity $O(n^2)$.

$$S_t = \begin{cases} 0 & \text{if } n = 0 \\ f(s_1) & \text{if } n = 1 \\ MAX\left\{F(s_i, s_j), f(s_i), f(s_j)\right\} & \text{if } n >= 2 \end{cases} \quad (1)$$

Generally speaking, if a personal information item is exposed, it will not cause too much harm to a user. However, when multiple personal information items constitute an information set, it will pose a great threat to the user's privacy. Therefore, the sensitivity of the information set must be considered.

2) IDENTITY TRUST EVALUATION

We associate the device layer with the social layer in SCMEC. We map the trust relationships established in the social layer to the device layer, and when a task is sensitive to the user, we only select one of the resource providers trusted by the user to offload the task. Therefore, we need to finish the trust evaluation in online social networks.

At present, there has been a lot of work on trust computing in online social networks, which can be divided into three categories: structure-based models, interaction-based models, and hybrid models [26]. Yin *et al.* [27] proposed *AUTrust* trust evaluation model to evaluate social network relationships, which is a hybrid model that comprehensively considers three categories of trust factors. He assigned the same weight to each category of factor and then integrated it into a global trust value.

We adopt the trust evaluation mechanism proposed by Zhao and Pan [28] as our identity trust evaluation method, which includes structural factors and interactive factors. He pointed out that most of the trust evaluations in existing studies just quantify several trust related factors and integrate them into a trust value by setting a weight for each factor. For one thing, it is difficult to determine the weight value of each factor; for another, the evaluation result is a value, which is not intuitive to users. Consequently, he formalized trust evaluation in online social networks as a classification problem in machine learning.

To evaluate the trust level in online social networks, it is essential to construct an identity trust feature vector. In online social networks, m and n are two users. We suppose that m is an evaluator and n is a object evaluated by m. The mechanism combines structural and interactive factors to construct a trust feature vector $v(m, n)$. If n is a fan of m, n is called a follower of m and m is called a friend of n. The elements of the vector are shown in Tab. 1. For more details, please refer to Zhao's paper, which will not be repeated here.

Corresponding to task sensitivity, we also divide identity trust into three levels. The identity trust level of n given by m can be expressed as $It(m, n)$. $It(m, n) \in \{1, 2, 3\}$. The values from 1 to 3 indicate that the trust levels are low, medium, and high, respectively. After constructing the trust feature vector $v(m, n)$ from the above eight factors, we need to map it to a discrete trust level ranging from 1 to 3. We use $ITC(v(m, n))$ to express the mapping from a trust feature vector to a trust level, that is, $It(m, n) = ITC(v(m, n))$.

TABLE 1. Elements of identity trust feature vector.

Name	Interpretation
$NumFr(n)$	The number of n's friends
$NumFl(n)$	The number of n's followers
$FFRatio(n)$	The ratio of $NumFl(n)$ and $NumFr(n)$
$NumBiFl(n)$	The number of bidirectional followers
$BFRatio(n)$	The ratio of $NumBiFl(n)$ and $NumFl(n)$
$Sim(m, n)$	The similarity between m and n
$Inter(m, n)$	The number of interactions between m and n
$Dis(m, n)$	The distance from m to n in the graph

3) BEHAVIOR TRUST EVALUATION

Behavior trust evaluation is a method to evaluate providers' behavior performance, which is used to predict the effectiveness of services. Due to the new characteristics of mobile edge computing, there will be some misbehaving providers, which will disrupt offloading and lead to low offloading effectiveness. Therefore, we need to establish a trust evaluation mechanism for their behavior and performance in this process.

However, it is difficult to choose an appropriate weight for each factor that affects behavior evaluation. Therefore, we also perform the evaluation of behavior trust by utilizing a classification approach in machine learning.

To evaluate the behavior, we first need to construct a behavior trust feature vector. There are mainly two main principles that need to be considered. Firstly, the feature vector should include factors that are closely related to the behavior during the task offloading process. Secondly, it can be retrieved from the interactive data without too much computing effort. In mobile edge computing, we assume that j is an idle device and i is a device which has a task to offload. According to previous work and our research, the elements of the behavior trust feature vector $v(i, j)$ are listed in Tab. 2.

To evaluate the behavior of j, we take 7 related factors into consideration. The first five factors are related to the performance of all services provided by j, and the last two factors are related to the performance of services provided by j for i. If we take a time interval t as a segment, we update the behavior trust level once per segment, and take the performance of j in the latest s segments as the data source.

For the first five features, $ANum(j)$ and $CNum(j)$ respectively correspond to the total number of services provided by j and the number of tasks successfully completed by j, while $ACRatio(j)$ stands for the ratio of $CNum(j)$ and $ANum(j)$. $Reputation(j)$ represents the overall impression of j given by all objects interacting with j, and $Reputation(j) \in [0, 1]$. ACT_p (actual completion time) and ECT_p (estimated completion time) respectively correspond to the actual completion time

TABLE 2. Elements of behavior trust feature vector.

Name	Interpretation
$ANum(j)$	The number of times that j provides services
$CNum(j)$	The number of times that j successfully completed these tasks
$ACRatio(j)$	The ratio of $CNum(j)$ and $ANum(j)$
$Reputation(j)$	Average score of j given by all objects interacting with j
$TD\_rate(j)$	Time deviation rate of j's services
$Num(i,j)$	The number of times that j provides services for i
$Score(i,j)$	The average score of j given by i

and the estimated completion time of a successfully completed task T_p. Equation (2) is used to calculate $TD\_rate(j)$. $TD\_rate(j)$ represents the average deviation rate of the actual time spent by j and the estimated time. By considering the time deviation rate, we can judge whether j is misbehaving in the process of providing services, such as reducing a large number of resources and not providing resources. For the last two factors, $Num(i,j)$ describes the number of times j provides services for i. $Score(i,j)$ describes the average score of j given by i according to j's services. It is noteworthy that we assume that all objects are willing to contribute their resources to each other.

$$TD\_rate(j) = \frac{1}{CNum(j)} \sum_{p=1}^{CNum(j)} \frac{(ACT_p - ECT_p)}{ECT_p} \quad (2)$$

According to actual requirements, we divide behavior trust into three levels. The values from 1 to 3 indicate that the trust levels are poor, acceptable, and strongly recommended, respectively. The first level is used to punish the resource providers for poor performance. If a resource provider's behavior trust level is 1, its behavior is unreliable and it is not recommended to be selected to perform a task. If a resource provider's behavior trust level is 2, it means that the behavior of the resource provider is relatively reliable and can be recommended to perform general tasks. Level 3 indicates that the behavior of the resource provider is highly reliable and it can be recommended to perform very important tasks that are not allowed to fail or are very sensitive to latency. The number of behavior trust levels can be further extended if needed. After constructing the feature vector, we need to map it to a discrete trust level ranging from 1 to 3. The behavior trust level of j given by i can be expressed as $Bt(i,j)$. We use $BTC(v(i,j))$ to express the mapping from the trust feature vector to a behavior trust level, that is, $Bt(i,j) = BTC(v(i,j))$.

IV. TRUST-AWARE TASK OFFLOADING FRAMEWORK

To reduce latency or energy consumption and ensure the trustworthiness of the services, we proposed a trust-aware task offloading framework (*TATOF*) in MEC, as shown in Fig. 2.

Fig. 2 illustrates the overall structure of our framework. There are mainly three modules, namely, trust evaluation module, filtering module, and selection module. UID_i is the relevant identification information of device i. Identity model and behavior model in the trust evaluation module

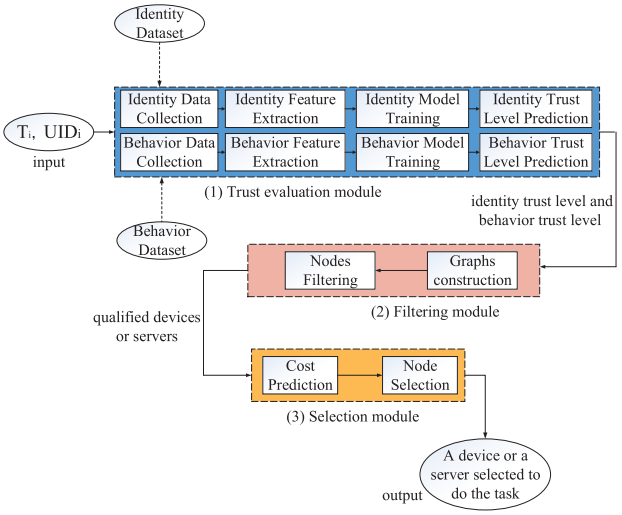


FIGURE 2. Trust-aware task offloading framework.

are completed in the training phase. Based on these two models, the identity trust and behavior trust levels of resource providers can be predicted.

The trust evaluation module gives the predicted behavior and identity trust levels of each provider and passes the predicted results to the filtering module. Next, the filtering module filters out the unqualified providers whose behavior trust level is lower than that required by the user and whose identity trust level is lower than task sensitivity level. This step guarantees the trustworthiness of services by removing these unqualified providers. Then, the filtering module hands over the qualified providers to the selection module. Last, the selection module calculates the cost of time and energy overhead and selects an appropriate resource provider for the task. It is important to note that we trust the behavior and identity of the MEC servers provided by operators by default.

A. TRUST EVALUATION

Our trust evaluation of resource providers includes two aspects: identity trust and behavior trust, which can be expressed as $Tr = (It, Bt)$. Identity trust (It) comes from the relationships that the owners of these devices establish at the social layer. Behavior trust (Bt) refers to the evaluation of resource providers' behavior and performance over a period of time.

We choose support vector machines (SVM) for classification. In machine learning, support vector machines are supervised learning models with associated learning algorithms for classification and regression analysis. There are two reasons why we choose SVM. On the one hand, since there is no labeled dataset for this research, we have to label the relevant data as dataset. Compared with other classification algorithms, SVM requires a relatively small sample size, which reduces our work on labeling data. On the other hand, SVM can not only achieve good results in processing linearly separable sample data, but also be good at dealing

with linearly inseparable sample data. We cannot guarantee that our labeled training data must be linearly separable.

The trust evaluation module is mainly composed of data collection, feature extraction, model training, and trust level prediction. In the data collection part, the behavior data is obtained through a simulation tool. We obtain L records in the format ($UIDB_i$, $UIDB_j$) generated by the simulation tool, and label these records with different levels. After that, the feature extraction sub-module will use $UIDB_i$ and $UIDB_j$ as parameters to produce L pairs (vector($UIDB_i$, $UIDB_j$), $Bt(i, j)$). Then, these data will be sent to the model training sub-module and used as training data. After the model is trained and tested, the trust level prediction sub-module will predict the trust level of any user pair on the basis of the trained model. The identity data is captured from a provided API of a social network application, and the processing steps are similar to behavior trust. We will not repeat it here.

In general, the SVM-based trust evaluation process can be summarized as follows.

- Collect a large number of trust feature vectors with the corresponding trust levels, and put them as training data.
- Choose suitable parameters and kernel function for SVM and train the classifier with training data.
- Test and validate our classifier. If the accuracy meets our requirements, then proceed to the next step, or go back to do some adjustment.
- Use the classifier to make inference, that is, when a feature vector $v(i, j)$ is given, the classifier can infer which trust level i holds on j.

B. FILTERING

The filtering module is mainly used to filter out unqualified nodes. If the task sensitivity level is higher than a provider's identity trust level, the provider is unqualified. If the behavior trust level of a provider is lower than the user's required behavior level, the provider is unqualified.

For the convenience of description, a computation task model is constructed here. For a computation task, we use a five-tuple $\langle I_i, L_i, O_i, S_i, B_i \rangle$ to describe it. I_i is the input data size of the task, which includes program code, input parameters, and so on. L_i is the amount of computing resource (i.e., number of CPU cycles) required by the task. O_i is the output data size of the task. S_i represents the sensitivity level of the task, which is obtained by analyzing the sensitivity of the sensitive information set contained in the task. B_i represents the behavior trust level required by the user generating the task.

To establish the relationship between the candidate resource providers and the devices generating tasks, we need to construct three related graphs, which are described in detail as follows.

Specifically, we introduce the device identity trust graph G^{tru} to model the social tie among the devices, as shown in Fig. 3. Each node represents a device. If there is a directed edge from node i to node j, it indicates that the owner of device i has an identity trust relationship with the owner of device j, and the weight of the directed edge represents the level of identity trust. In particular, when a device and another device belong to the same user, the weight of the directed edge between the two devices is 3.

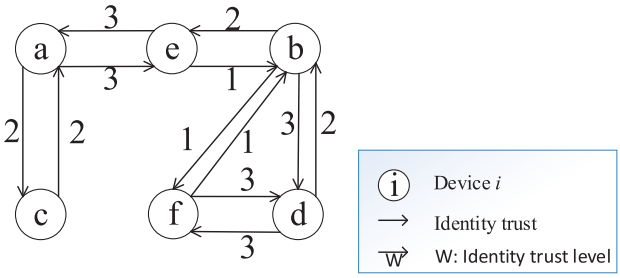


FIGURE 3. Device identity trust graph.

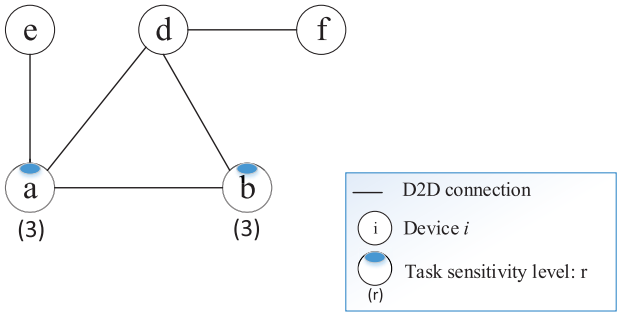


FIGURE 4. D2D connection graph.

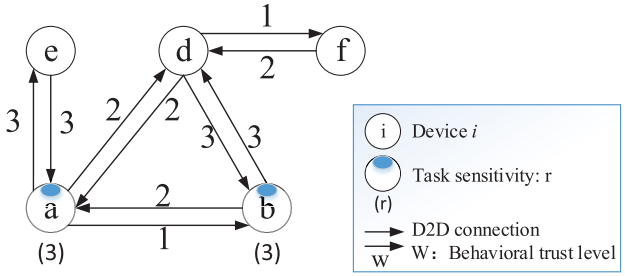


FIGURE 5. D2D connection associated with behavior trust graph.

Next, we introduce the D2D connection graph and the D2D connection associated with behavior trust graph. Based on the technology of D2D communication, we can acquire the devices in the current D2D communication range in real time. These devices within the communication range can form a connectable graph, which is a D2D connection graph, as shown in Fig. 4. Each node represents a device. If there is a connection between two devices, it means that the two devices are in the range of communication with each other. If there is a small blue ellipse in a node, it indicates that a task is generated on the device node. The small ellipse has a task sensitivity attribute S_i which indicates that the sensitivity level of the task is r. On the basis of Fig. 4, we introduce behavior trust between devices, and form a D2D connection associated with behavior trust graph, as shown in Fig. 5. If there is a directed edge from node i to node j, the weight of the edge represents the behavior trust level of j given by i.

After constructing the above diagrams, we can perform filtering in the nodes filtering sub-module to remove those unqualified nodes.

For a task $T_i = \langle I_i, L_i, O_i, S_i, B_i \rangle$ on device i, the filtering module will remove the nodes that can not meet the requirements of S_i and B_i. S_i determines the required identity trust level, while B_i determines the required behavior trust level. According to whether the task contains sensitive information, there are two cases.

In one case, when the task does not contain any sensitive information, we think that the sensitivity level of the task is 0. We no longer need to consider identity trust. For a task T_i on device i, the qualified node j in the array $d2dTrust[\,]$ must satisfy the following requirement: In Fig. 5, there is an edge from i to j and the weight of the edge is greater than or equal to B_i (i.e., $B_i <= Bt(i,j)$). In this case, the qualified nodes in the array $d2dTrust[\,]$ are within the range of its D2D communication and meet the required behavior trust level at the same time.

In the other case, when the task contains sensitive information, the sensitivity level of the task is S_i. We need to combine Fig. 3 and Fig. 5 for filtering. For a task T_i on device i, the qualified node j in the array $d2dTrust[\,]$ must satisfy the following requirements: In Fig. 5, there is an edge from i to j and the weight of the edge is greater than or equal to B_i (i.e., $B_i <= Bt(i,j)$); in Fig. 3, there is an edge from i to j and the weight of the edge is greater than or equal to the S_i of the task in Fig. 5 (i.e., $S_i <= It(i,j)$).

We give the corresponding trust-based filtering algorithm 1 as follows.

C. SELECTION

Qualified nodes after filtering are considered to be trusted by the user both in identity and behavior. For one thing, they will not illegally obtain our sensitive information; for another, they behave well during the task offloading process.

Faced with a large number of candidate nodes, how to select a corresponding node to complete the task is generally determined by an offloading policy. In an offloading policy, there may be only one execution mode or several optional execution modes. In this module, we adopt a hybrid offloading policy, which mainly provides four alternative modes of task execution, as shown in Fig. 6. This policy mainly refers to the work of Chen *et al.* [29] and we expand latency on this basis. The four modes it provides are described as follows.

- Local execution: Users can choose to perform tasks on their mobile devices without the overhead of task offloading.
- D2D offloaded execution: Devices in proximity at the network edge can share their computing resources through D2D communication to help other devices complete their tasks, and these devices can benefit from the mode.
- Direct edge server offloaded execution: A device can offload its tasks directly to an edge server.

Algorithm 1 Trust-Based Filtering Algorithm 1

Input: a task T_i, task's id: *task_id*
Output: $d2dTrust[\,]$

```
get device number device_i via task_id
get the position location_i of devicei
get the array d2dConnection[ ] of devices within the D2D communication range of devicei via location_i
s_i =getTaskSensitivity(T_i)
b_i=getBehaviorLevel(T_i)
if s_i=0 then
   for each d2d_j in d2dConnection[ ] do
      Bt=getBehaviorTrust(device_i, d2d_j)
      if Bt ⩾ b_i then
         d2dTrust [counter]=d2d_j
         counter++
      end if
   end for
else
   get the owner deviceIOwner of device i by device_i
   get the array social_person[ ] of human who have social relationship with deviceIOwner
   trustPerson[ ]=returnTrustPerson(s_i, social_person[ ])

   for each d2d_j in d2dConnection[ ] do
      Bt=getBehaviorTrust(device_i, d2d_j)
      if Bt ⩾ b_i then
         deviceJOwner=getOwnerOfDevice(d2d_j)
         if deviceIOwner==deviceJOwner then
            d2dTrust [counter]=d2d_j
            counter++
         end if
         for each trust_id in trustPerson[ ] do
            if deviceJOwner==trust_id then
               d2dTrust [counter]=d2d_j
               counter++
            end if
         end for
      end if
   end for
end if
return d2dTrust [ ]
```

- D2D-Assisted edge server offloaded execution: A device with poor cellular connection first transmits its computation task to a neighboring device through D2D communication. The neighboring device has a strong cellular connection, which can help it to offload the task to an edge server for execution. After the task is completed, the output can be obtained through the neighboring device.

It is noteworthy that we think for users from users' point of view to minimize the consumption of all users in the system. Edge servers (also known as MEC servers) generally have constant power supply. Therefore, we do not add the energy consumption of MEC servers to the cost of execution modes.

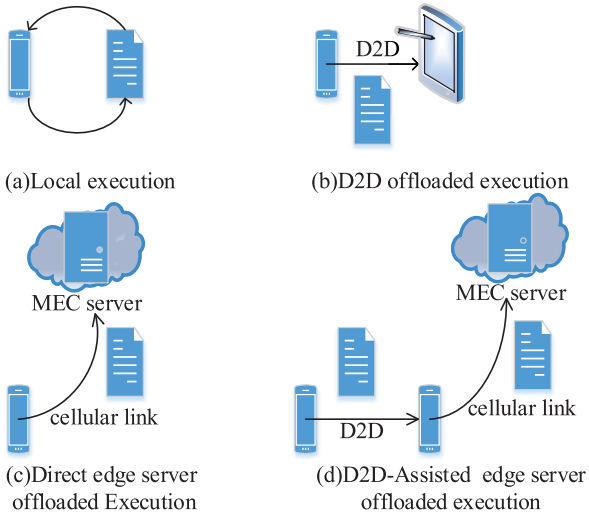


FIGURE 6. An illustration of four different task execution modes.

According to the task five-tuple $\langle I_i, L_i, O_i, S_i, B_i \rangle$, the latency and energy consumption in the above four different execution modes can be given as follows, which are realized by the cost prediction sub-module.

- Local execution: In this mode, the time spent mainly comes from the task's local computation time. The execution time for computation is given by $T_i^l = L_i/c_i$ and the energy consumption is given by $E_i^l = \rho_i^c L_i$. Among them, $c_i = (1-\delta_i) Z_i$ represents the CPU computing power provided by device i per unit time and Z_i represents the CPU working frequency. Besides, δ_i represents the current load ratio (device i may need to run some background loads) and ρ_i^c is the energy cost per CPU cycle for computation. Accordingly, we can obtain the cost of the local execution mode in terms of both time and energy overhead as $\theta_i^l = \lambda_i^t T_i^l + \lambda_i^e E_i^l$. $\lambda_i^e, \lambda_i^t \in [0, 1]$ denote the weights of time and energy for device i's selection and $\lambda_i^e + \lambda_i^t = 1$. The user can adjust the two weight parameters according to different situations. For example, when a user is running some applications that are sensitive to latency, the user can set $\lambda_i^e = 0$ and $\lambda_i^t = 1$ in the selection process. In contrast, when a device is at a low battery state and the user cares about energy consumption, the user can set $\lambda_i^e = 1$ and $\lambda_i^t = 0$.
- D2D offloaded execution: In this mode, the cost of time and energy is mainly derived from the transmission of a task and the execution of the task. The time spent is given by $T_{ij}^{d1} + T_{ij}^{d2}$ and the energy consumption is given by $E_{ij}^{d1} + E_{ij}^{d2}$. In this case, the time spent in transmitting input and output data between two devices is given by $T_{ij}^{d1} = I_i/D_{ij} + O_i/D_{ji}$ and energy consumption of transmitting input and output data between two devices is given by $E_{ij}^{d1} = \left(P_{it}^d + P_{jr}^d\right) I_i/D_{ij} + \left(P_{jt}^d + P_{ir}^d\right) O_i/D_{ji}$. The time for executing the offloaded task by device j is given by $T_{ij}^{d2} = L_i/c_j$ and the energy for executing the offloaded task by device j is given by $E_{ij}^{d2} = \rho_j^c L_i$. Among them, P_{it}^d and P_{ir}^d represent the energy consumption of device i in transmitting and receiving data via D2D per unit time, respectively, and D_{ij} represents the rate when data is transmitted from i to j through D2D. c_j and ρ_j^c are similar to those mentioned in local execution. Accordingly, we can obtain the cost of the D2D offloaded execution mode in terms of both time and energy overhead as $\theta_{ij}^d = \lambda_i^t \left(T_{ij}^{d1} + T_{ij}^{d2}\right) + \lambda_i^e \left(E_{ij}^{d1} + E_{ij}^{d2}\right)$.
- Direct edge server offloaded execution: In this mode, the cost of energy for device i is mainly derived from the transmission of the task, and the cost of time is mainly derived from transmission and execution of the task. The time consumption is given by $T_i^{c1} + T_i^{c2}$ and the energy consumption is given by E_i^c. In this case, the time consumption of transmitting input and output data between the device and an edge server is given by $T_i^{c1} = I_i/D_i^t + O_i/D_i^r$ and energy consumption of transmitting input and output data between the device and an edge server is given by $E_i^c = P_{it}^c I_i/D_i^t + P_{ir}^c O_i/D_i^r$. The time for executing the offloaded task in the edge server is given by $T_i^{c2} = L_i/K_m$. Among them, D_i^t and D_i^r represent the speed of uploading and downloading data between i and the edge server, respectively, and K_m represents the CPU operating frequency of the virtual machine m assigned to the task by the server. Accordingly, we can obtain the cost of the Direct edge server offloaded execution mode in terms of time and energy overhead as $\theta_{im}^c = \lambda_i^t \left(T_i^{c1} + T_i^{c2}\right) + \lambda_i^e E_i^c$.
- D2D-Assisted edge server offloaded execution: In this mode, the cost of energy for devices is mainly derived from the transmission of a task, and the cost of time is mainly derived from the transmission and execution of the task. The time consumption is given by $T_{ij}^{dc1} + T_j^{dc2}$ and the energy consumption is given by $E_{ij}^{dc1} + E_j^{dc2}$. In this case, the time consumption of transmitting input and output data is given by $T_{ij}^{dc1} = I_i/D_{ij} + O_i/D_{ji} + I_i/D_j^t + O_i/D_j^r$. The time consumption of task execution is given by $T_j^{dc2} = L_i/K_m$. K_m is similar to hat mentioned in the above execution mode. The energy consumption of transmitting input and output data between two devices is given by $E_{ij}^{dc1} = \left(P_{it}^d + P_{jr}^d\right) I_i/D_{ij} + \left(P_{ir}^d + P_{jt}^d\right) O_i/D_{ji}$. Besides, the energy consumption of transmitting input and output data between device j and an edge server is given by $E_j^{dc2} = P_{jt}^c I_i/D_j^t + P_{jr}^c O_i/D_j^r$. Accordingly, we can obtain the cost of the D2D-Assisted edge server offloaded execution mode in terms of both time and energy overhead as $\theta_{ij}^{dc} = \lambda_i^t \left(T_{ij}^{dc1} + T_j^{dc2}\right) + \lambda_i^e \left(E_{ij}^{dc1} + E_j^{dc2}\right)$.

Based on the above analysis, we can predict the latency and energy consumption under different task execution modes, and select an appropriate mode for a user. However, when multiple users in the system have tasks waiting for offloading,

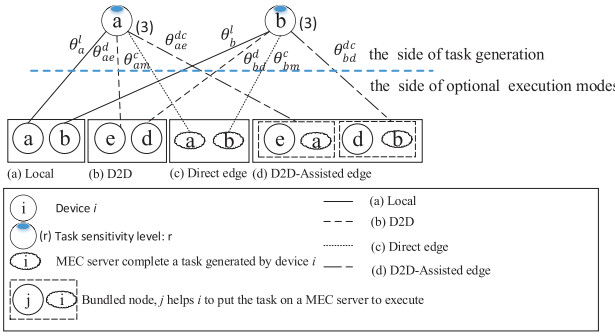


FIGURE 7. An illustration of bipartite graph of task execution modes.

how to choose an appropriate execution mode for each user? The node selection sub-module can also help us to solve this problem.

To face the multi-user scenario in MEC, the multi-user offloading problem is reduced to a matching problem of bipartite graph. To minimize the overhead of the whole system, we use the minimum-weight bipartite perfect matching solution over the bipartite graph to select a suitable node for each user device to complete a task.

The nodes left by the common action of the three graphs of the filtering module are qualified. If users request the behavior trust level of resource providers to be acceptable, a bipartite graph consisting of the candidate nodes (edge servers or qualified devices) and the tasks nodes waiting to select execution modes is shown in Fig. 7. The graph is divided into the side of task generation (referred to as the initial side) and the side of optional execution modes (referred to as the execution side). A node i of the initial side indicates that device i has a task, and a node j of the execution side represents an optional object for task offloading. The weight of one edge represents the time and energy consumption when a task chooses this execution mode. The construction method of the bipartite graph is given as follows. By using the minimum-weight bipartite perfect matching method, we can choose an appropriate node for each task. For our problem, the number of edges is proportional to the number of nodes N, and hence our bipartite matching method has a low complexity of $O(N^2)$, which can scale well for practical implementation.

- Local execution: There is an edge between a task node i and its corresponding local node i, and the weight of this edge is θ_i^l.
- D2D offloaded execution: There is an edge between a task node i and a node j if node j is in i's $d2dTrust[\,]$ after filtering for device i, and the weight of this edge is θ_{ij}^d.
- Direct edge server offloaded execution: There is an edge between a task node i and its subscribed server node, and the weight of this edge is θ_{im}^c.
- D2D-Assisted edge server offloaded execution: There is an edge between a task node i and a bundled node (j and the edge server serving for j) if node j is in i's $d2dTrust[\,]$ after filtering for device i, and the weight of this edge is θ_{ij}^{dc}.

V. PERFORMANCE EVALUATION

To verify the effectiveness of our proposed framework, we have carried out some experiments. We mainly used LIBSVM tool [30] for model training in trust evaluation module and EdgeCloudSim [31], [32] for simulating mobile edge computing environment. LIBSVM tool is an integrated toolkit developed for supported vector machine and has been widely used. EdgeCloudSim is a simulation tool that supports the modeling of both computational and networking resources to handle the edge computing scenarios and it has good extensibility. It includes task generation module, mobility module, network environment simulation module, offloading policy module and so on. User devices include smart bracelet, wearable glasses, wearable medical devices, mobile phones, tablets, computers, etc., covering a wide range. D2D connections between devices will change with movement of devices, which depends on mobility module in the EdgeCloudSim. Meanwhile, the cellular data rate decreases with the increase of the number of devices in the hotspot, because we simulate the network environment by this tool. We modified and extended this tool to test the effectiveness of our framework. In particular, we searched for answers to the following three research questions:

- RQ1: How does the classification of machine learning perform in trust evaluation (i.e., identity trust and behavior trust)?
- RQ2: After introducing behavior trust evaluation, can the misbehaving nodes be filtered out for users to improve the performance of task offloading?
- RQ3: What is the performance of our framework in reducing latency or energy consumption for users?

A. EXPERIMENTS ON RQ1

1) EXPERIMENTAL DESIGN OF PREDICTION PERFORMANCE

Firstly, we need to collect relevant data, and then label them with different levels. Secondly, after the dataset is ready, to give full play to its performance, SVM needs to select an appropriate kernel, the kernel's parameters, and soft margin parameter C. Since there may be linearly inseparable in trust evaluation, we choose the RBF kernel. The best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and the kernel's parameter γ. Finally, there are several measures commonly used in machine learning. For each category, a confusion matrix is shown in Tab. 3, where m_1 represents the number of items correctly predicted for the category, m_2 represents the number of items that should fall into this category but are predicted to fall outside that category, m_3 represents the number of items that do not belong to this category but are predicted to fall into this category, and m_4 represents the number of items that do not belong to this category and are correctly predicted. In our experiments, three measures are taken into consideration: precision, recall and F-measure. The precision P is $P = m_1/(m_1 + m_3)$. The recall R is $R = m_1/(m_1 + m_2)$, and the F-measure is defined as $F = 2PR/(P + R)$.

TABLE 3. Confusion matrix.

	Prediction 1	Prediction 0
Truth 1	m_1	m_2
Truth 0	m_3	m_4

Weibo, one of the most popular social networks in China, was selected as the data source for user identity trust evaluation. With the API of Weibo which is open for developers to build applications, a lot of information can be obtained, including user configuration, relationships, and so on.

In the data collection part of identity trust evaluation, the data we employ is originated from Weibo dataset of KDD-Cup2012 track one [33]. The dataset represents a sampled snapshot of Weibo, and provides standardized format and abundant information in multiple domains. There are 7 txt files in all. Our experiment makes use of the information in three of them which include: user_profile.txt, user_sns.txt and user_action.txt. Because we only consider the identity trust levels that one user has on its followers, $Dis(m, n)$ in the identity trust feature vector is set to be 1 here. To ensure the quality of training data, we randomly selected 1920 records and invited three experts in the field of trust evaluation to label these records at the level of 1 to 3. After labeling the records, we used 1728 records as training data and the remaining 192 records as test data. In the experiment of identity trust, we obtained the best accuracy under the combination of $C = 512$ and $\gamma = 8$. We compared our identity trust evaluation method with the *AUTrust* method of Yin et al. (refer to [27]). *AUTrust* is a trust evaluation method of social networks by assigning weights to trust factors. The trust factors in this method are basically the same as those in our identity trust evaluation method. The author divides these trust factors into three categories and assigns equal weight to each category.

In the data collection part of behavior trust evaluation, we modified and extended EdgeCloudSim to collect data that needs to be used. To ensure the quality of training data, we invited three experts who have studied mobile edge computing to label these data. We divided behavior trust into three levels, and labeled 1920 records, of which 1728 records were used as training data, and the remaining 192 records were used as test data. In the experiment of behavior trust, we obtained the best accuracy with the combination of $C = 32$ and $\gamma = 8$. We compared our behavior trust evaluation method with the *MSTrust* method of Yuan et al. (refer to [17]). *MSTrust* is a trust evaluation method for service behavior in edge computing and the author gives a formula for aggregating global trust. This method mainly considers three trust factors: success rate, direct feedback and overall reputation.

2) THE RESULT

RQ1: How does the classification of machine learning perform in trust evaluation (i.e., identity trust and behavior trust)?

TABLE 4. Measurement of identity trust classifier.

	P	R	F
Level 1 (low)	0.9302	0.9523	0.9411
Level 2 (medium)	0.9268	0.9268	0.8589
Level 3 (high)	0.8461	0.8461	0.7159

TABLE 5. Measurement of behavior trust classifier.

	P	R	F
Level 1 (poor)	0.9393	0.9117	0.9252
Level 2 (acceptable)	0.94	0.94	0.8836
Level 3 (strongly recommended)	0.8461	0.9166	0.8799

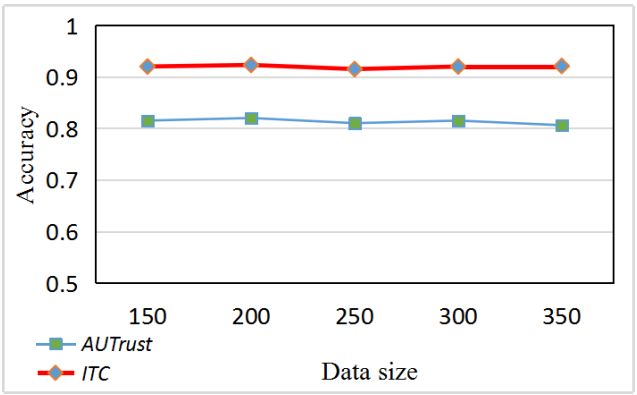


FIGURE 8. Changes in accuracy of identity trust.

Tab. 4 shows the results of three measures mentioned above for each level in identity trust evaluation, and Tab. 5 shows the results of three measures mentioned above for each level in behavior trust evaluation. They verify the usability and good performance of the trust classifiers we trained.

To verify the accuracy of the classifiers we trained, we added some data and have performed experiments on different test data sizes.

The comparison result of identity trust evaluation is shown in Fig. 8. *ITC* is our identity trust classifier. To compare our identity classifier and *AUTrust*, the experts analyzed the global trust value and divided the range of [0, 1] into three intervals [0, 0.3), [0.3, 0.6), and [0.6, 1] according to their experience. These three intervals correspond to level 1, 2 and 3 of identity trust respectively. As shown in the figure, our classifier can achieve better accuracy. Because in the *AUTrust* method, each category of trust factor is given the same weight, which is inaccurate and unrealistic.

The comparison result of behavior trust evaluation is shown in Fig. 9. *BTC* is our identity trust classifier. To compare our behavior classifier and *MSTrust*, the experts analyzed the global trust value and divided the range of [0, 1] into three intervals [0, 0.35), [0.35, 0.6), and [0.6, 1] according to their experience. These three intervals correspond to level 1, 2 and 3 of behavior trust respectively. As shown in the figure, our classifier can achieve better accuracy. Because in the *MSTrust* method, only three trust factors, namely success rate, direct feedback and overall reputation, are considered.

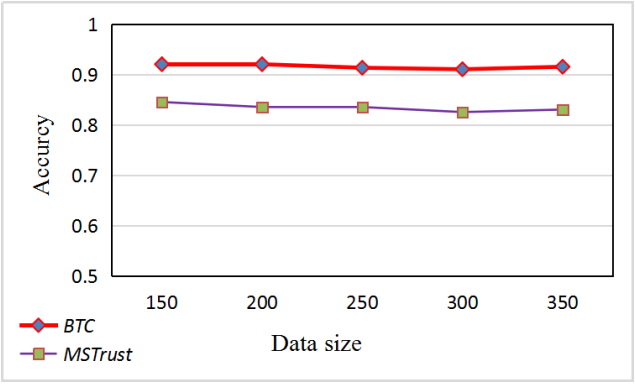


FIGURE 9. Changes in accuracy of behavior trust.

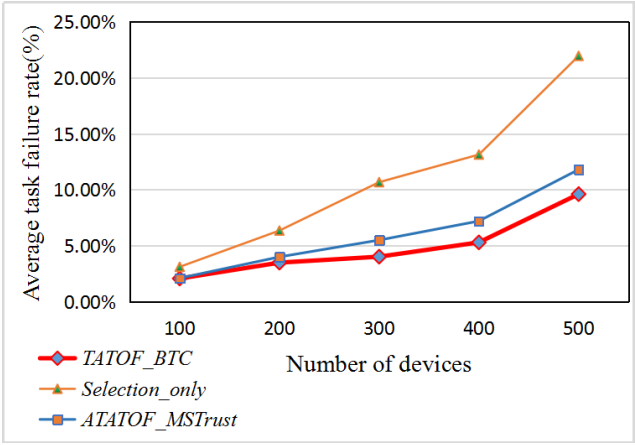


FIGURE 10. Average task failure rate.

Our behavior trust evaluation method includes not only the above trust factors, but also the total number of providing services (i.e., ANum(j)) and the time deviation rate (i.e., TD_rate(j)). These two trust factors can help us better evaluate the behavior of resource providers. For example, one provider provided and successfully completed 10 times service while the other provided and successfully completed 90 times service. Although both providers have a 100% success rate, we have a higher trust level for the provider that completed service 90 times. The time deviation rate can help us identify whether the provider reduces a large amount of resources in the service process without agreement.

B. EXPERIMENTS ON RQ2

1) EXPERIMENTAL DESIGN OF THE IMPACT AFTER INTRODUCING TRUST EVALUATION

We invited three experts in edge computing to inject several misbehaving devices that reduce a lot of resources or intentionally fail to perform tasks correctly during service into the simulation tool. When tasks are assigned to these devices, they may fail because of their improper behavior. By introducing our behavior trust evaluation, we can filter out these misbehaving devices and reduce the failure rate of tasks in the system. We compared the task failure rate in three cases: our *TATOF* using *BTC* trust classifier (i.e., *TATOF_BTC*), only selection module (i.e., *Selection_only*), and another *ATATOF* that replaces our *TATOF*'s behavior trust evaluation mechanism with *MSTrust* (i.e., *ATATOF_MSTrust*).

2) THE RESULT

RQ2: After introducing behavior trust evaluation, can the misbehaving nodes be filtered out for users to improve the performance of task offloading?

The average task failure rate in the system is shown in Fig. 10. The *Selection_only* indicates that it has only selection module but no trust evaluation and filtering modules. In this case, task failure rate is high because of some misbehaved nodes. When behavior trust evaluation is introduced, the nodes with poor performance are filtered out, so the task failure rate of the system is reduced. At the same time, our *TATOF_BTC* has a lower task failure rate than that of *ATATOF_MSTrust* because our behavior trust evaluation mechanism takes more behavior factors into account and can better identify some improper behaviors. Task failure occurs in two cases. In one case, when misbehaving devices incorrectly complete some tasks, these tasks will fail. In the other case, because this paper does not deal with the mobility management of task migration, we also assume that a task will fail when two devices are out of communication range with each other. Task migration means that the virtual machine shoule be migrated if a user roam from one network region to another. In fact, we will do further research on task migration in the future. When a device is beyond the original communication range due to mobility, we will choose an appropriate approach to transmit output data to the user to ensure the continuity of service.

C. EXPERIMENTS ON RQ3

1) EXPERIMENTAL DESIGN OF REDUCING LATENCY AND ENERGY CONSUMPTION

The main purpose of this experiment is to verify the performance of the proposed task offloading framework in reducing latency and energy consumption as the number of devices increases. After adding task sensitivity computation, trust evaluation and filtering, these processes will introduce some time overhead. In this case, can our framework reduce latency or energy consumption for users during task offloading? To better observe the effect of our framework, we set up three offloading scenarios (i.e., local computation, only offloaded to the edge server (*only_edge*), and our proposed *TATOF*). The average reduction rates of time and energy of another two scenarios are compared by using the local computation as a benchmark.

The levels of identity and behavior trust between devices is invariable in this experiment. Tab. 6 shows the simulation settings, most of which are consistent with the actual measured values in practice [34], [35]. The number of computing resources required by a task is expressed in terms of the MIPS and 1Mips = 4MHz.

TABLE 6. Simulation setting.

Category	Parameter	Value/Range
Cellular Link	Cellular Data Rate	$[1, 12]$Mbps
	Cellular Power	600mW
User&Task	CPU Frequency	$[0.8, 4.4]$ GHz
	CPU Power	$[800, 1200]$mW
	CPU Load	$[0, 10\%]$
	Amount of Uploaded Data	600KB
	Amount of Download Data	500KB
	Length of the task	1000Mips
D2D Link	Maximum D2D Distance	200m
	Maximum D2D Bandwidth	20Mhz
	D2D Power	200mW

For better observation, when we focus on the performance of reducing latency, we assume that users' weight of energy consumption in task offloading is 0. Conversely, when we focus on the performance of reducing energy consumption, we assume that users' weight of latency is 0.

2) THE RESULT

RQ3: What is the performance of our framework in reducing latency or energy consumption for users?

In Web services, users need to provide some personal information required by services as input when enjoying services. Similar to Web services, a task generated by an application needs to collect some personal information as input to complete a specific function. The information items that need to be collected are predetermined, predefined, and limited in number. For the computation of task sensitivity, if there are n personal information items needed in a task, the number of relevant sensitivity composed of each two personal information items is C_n^2. Therefore, combined with the analysis of the task sensitivity formula, the computation of task sensitivity has a low complexity of $O(n^2)$. For the trust evaluation, once the model is completed in the training phase, the trust classifier is obtained. In practice, the time cost of predicting trust level based on the classifier is low. For the filtering process, if the number of devices in the communication range of device i is N, the process has a low complexity of $O(N^2)$ because the identity trust level and behavior trust level of these N devices need to be considered at the same time. Therefore, our framework does not cause too much time overhead.

As the number of devices in the system increases, the average time reduction rate of the two scenarios is shown in Fig. 11. When the number of devices in the system increases gradually, the cellular data rate between a device in a hotspot and an edge server decreases, and the transmission time increases. At the same time, as the number of devices increases, edge servers have to handle more tasks, so the servers' response time to tasks increases. Due to the above two reasons, the time reduction rate of *only_edge* is gradually reduced, and our *TATOF*, with the help of D2D and D2D-Assist execution modes, can still maintain a better performance than *only_edge*.

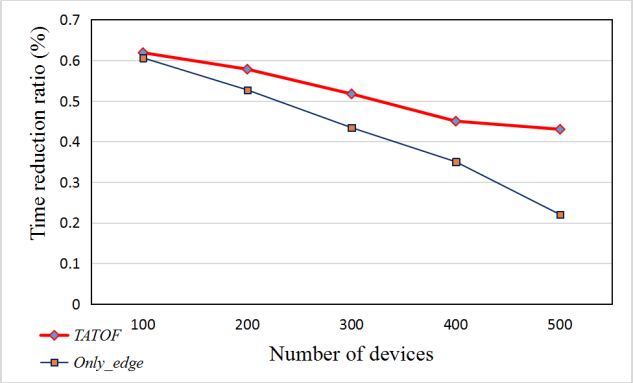


FIGURE 11. Time reduction rate.

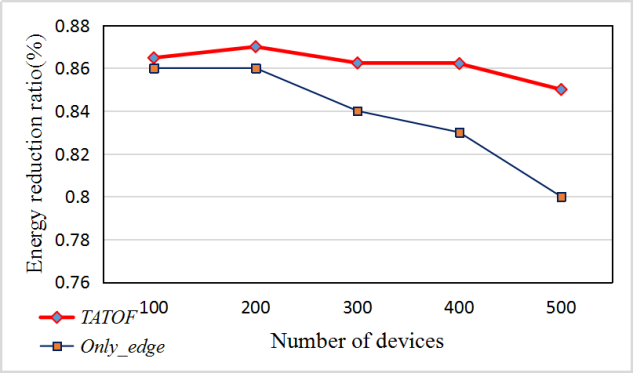


FIGURE 12. Energy reduction rate.

The average energy reduction rate of the two scenarios is shown in Fig. 12. As the number of devices increases, the number of alternative devices for D2D connections increases, so the energy reduction rate increases a little under *TATOF*. As the number of devices continues to grow, the cellular data rate between devices in a hotspot and edge servers decreases. As a result, transmission time increases and energy reduction rate in the system decreases. Under *TATOF*, with the help of D2D and D2D-Assist execution modes, the energy reduction rate is superior to that of *only_edge* scenario.

In conclusion, although our *TATOF* takes into account trust evaluation mechanism and filters out unqualified resource providers to ensure service trustworthiness, it can still reduce latency or energy consumption for users.

VI. CONCLUSION AND FUTURE WORK

We introduce online social networks into mobile edge computing and propose a trust-aware task offloading framework. To ensure the trustworthiness of services, we introduce identity trust and behavior trust into trust evaluation in mobile edge computing. To address the challenge that it is difficult to assign an appropriate weight to each trust factor, we formalize trust evaluation of mobile edge computing into a classification problem. Then, we filter out unqualified nodes that do not meet the requirements of tasks. Finally, the selection module selects one of the qualified providers for a user to execute the task based on the offloading policy. Based on our framework,

we are able to reduce latency or energy consumption and ensure the trustworthiness of the services for users.

In future work, we plan to add more related factors into a trust feature vector and conduct experiments on larger dataset. Meanwhile, we will conduct further research on mobility management of task migration. When a device is out of the original communication range due to movement, we want to choose an appropriate way to return the output data of the task to the device.

REFERENCES

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[2] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. MobiSys*, San Francisco, CA, USA, 2010, pp. 49–62.

[3] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[5] F. Bonomi, R. A. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC SIGCOMM*, Helsinki, Finland, 2012, pp. 13–16.

[6] J. Zhang, B. Chen, and Y. Zhao, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[7] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma, and J. Ott, "Security and privacy in device-to-device (D2D) communication: A review," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1054–1079, 2nd Quart., 2017.

[8] R. Roman, J. López, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comp. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[9] G. Shang-Fu and Z. Jian-Lei, "A survey of reputation and trust mechanism in peer-to-peer network," in *Proc. Int. Conf. Ind. Control Electron. Eng.*, Aug. 2012, pp. 116–119.

[10] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[11] W. Li, J. Cao, K. Hu, J. Xu, and R. Buyya, "A trust-based agent learning model for service composition in mobile cloud computing environments," *IEEE Access*, vol. 7, pp. 34207–34226, 2019.

[12] X. Lin, R. Lu, X. Liang, and X. Shen, "STAP: A social-tier-assisted packet forwarding protocol for achieving receiver-location privacy preservation in VANETs," in *Proc. IEEE Comput. Commun. Soc.*, Shanghai, China, Apr. 2011, pp. 2147–2155.

[13] M. Hussain and B. M. Almourad, "Trust in mobile cloud computing with LTE-based deployment," in *Proc. IEEE 14th Int. Conf. Commun. Assoc. Workshops*, Bali, Indonesia, Dec. 2014, pp. 643–648.

[14] W. Ahmad, S. Wang, A. Ullah, Sheharyar, and Z. Mahmood, "Reputation-aware trust and privacy-preservation for mobile cloud computing," *IEEE Access*, vol. 6, pp. 46363–46381, 2018.

[15] P. Zhou, K. Wang, J. Xu, and D. Wu, "Differentially-private and trustworthy online social multimedia big data retrieval in edge computing," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 539–554, Mar. 2019.

[16] J. Xu, S. Wang, B. K. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3538–3547, Jun. 2019. doi: 10.1109/TII.2019.2896965.

[17] J. Yuan and X. Li, "A multi-source feedback based trust calculation mechanism for edge computing," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr. 2018, pp. 819–824.

[18] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[19] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. ICC*, Paris, France, May 2017, pp. 1–6.

[20] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," *Sensors*, vol. 19, no. 5, p. 1105, Mar. 2019.

[21] C. Dong and W. Wen, "Joint optimization for task offloading in edge computing: An evolutionary game approach," *Sensors*, vol. 19, no. 3, p. 740, Feb. 2019.

[22] K. Li, M. Tao, and Z. Chen, "Exploiting computation replication for mobile edge computing: A fundamental computation-communication tradeoff study," 2019, *arXiv:1903.10837*. [Online]. Available: https://arxiv.org/abs/1903.10837

[23] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[24] *Standards for Privacy of Individually Identifiable Health Information*, Standard 45 CFR Parts 160 and 164, HIPAA, Dec. 2000.

[25] J. R. Weible, "Privacy and data: An empirical study of the influence of types of data and situational context upon privacy perceptions," Ph.D. dissertation, Dept. Elect. Eng., Mississippi State Univ., Starkville, MS, USA, 1993.

[26] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 47:1–47:33, Aug. 2013.

[27] G. Yin, F. Jiang, S. Cheng, X. Li, and X. He, "AUTrust: A practical trust measurement for adjacent users in social networks," in *Proc. CGC*, Hunan, China, Nov. 2012, pp. 360–367.

[28] K. Zhao and L. Pan, "A machine learning based trust evaluation framework for online social networks," in *Proc. IEEE TrustCom*, Beijing, China, Sep. 2014, pp. 69–74.

[29] X. Chen, Z. Zhou, W. Wu, D. Wu, and J. Zhang, "Socially-motivated cooperative mobile edge computing," *IEEE Netw.*, vol. 32, no. 6, pp. 177–183, Nov./Dec. 2018.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[31] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, Aug. 2018, Art. no. e3493.

[32] *Simulation Tools: EdgeCloudSim*. Accessed: Oct. 16, 2019. [Online]. Available: http://github.com/CagataySonmez/EdgeCloudSim

[33] *Kddcup*. Accessed: Oct. 16, 2019. [Online]. Available: http://www.kddcup2012.org/c/kddcup2012-track1

[34] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE ICC*, Paris, France, May 2017, pp. 1–6.

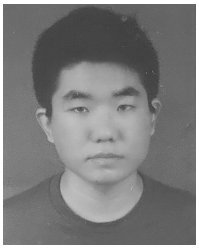
[35] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.

DEXIANG WU received the B.S. degree in computer science from Jiangsu Normal University. She is currently pursuing the M.S. degree with the Computer Science Department, Nanjing University of Aeronautics and Astronautics, Nanjing. Her current research interests include mobile edge computing, cloud computing, access control, privacy preservation, and service composition.

GUOHUA SHEN received the M.S. and Ph.D. degrees in computer science from the Nanjing University of Aeronautics and Astronautics, China. He is currently an Associate Professor with the College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics. His research interests include requirement traceability, fog computing, description logic, semantic web, and web services and ontology.

ZHIQIU HUANG received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics. He is currently a Professor with the College of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics. He is also the Director of software safety in computer science with the Nanjing University of Aeronautics and Astronautics. His research interests include formal method, cloud computing, edge computing, web security, and privacy preservation.

YAN CAO is currently pursuing the Ph.D. degree with the Computer Science Department, Nanjing University of Aeronautics and Astronautics, Nanjing. Her research interests include formal method, access control, cyber physical systems, information security and privacy preservation, and mobile edge computing.



TIANBAO DU is currently pursuing the M.S. degree with the Computer Science Department, Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests include requirement traceability, machine learning, software engineering, and edge computing.