ABSTRACT

Mobile edge computing is playing an increasingly important role in the rise of mobile internet technology. Services deployed on edge servers nearby mobile users would provide computing capabilities with low latency and high scalability. Usually, a single service is challenging to meet a complex user request, which asks for composing services. With the increasing number of services in the cloud and edge computing environment and the user mobility, selecting appropriate services to meet the complex mobile user’s requests becomes a crucial problem. This paper proposes a modified moth-flame optimization algorithm using overall QoS for service selection. Specifically, the overall QoS of services is calculated by combining the subjective and objective QoS with the ordinal relationship and coefficient of variation, and the moth-flame optimization algorithm is improved by adding the differential evolution algorithm. The experimental results show that the proposed approach outperforms some other services selection approaches.

KEYWORDS

Cloud Computing, Edge Computing, Mobile Service, Moth-Flame Optimization, QoS, Service Selection

INTRODUCTION

With the rapid development of mobile Internet technology, service computing in the mobile environment has become a hot research field in recent years. Specifically, with the emergence of edge computing, services can be deployed closer to mobile users to provide corresponding functionalities. Compared with traditional cloud computing, services deployed in the edge computing environment can effectively reduce the distance to users, ensure efficient network operation delivery, and realize the service interaction experience of high performance, low latency, and high bandwidth (Corcoran et al., 2016). However, due to the mobile characteristics of users and the limited resources of edge devices, the services provided by edge devices alone cannot meet a large number of increasingly complex computing requirements (Personè & Grassi, 2019). Cloud computing is good at global, non-real-time, and long-term computing, while edge computing is more suitable for local, real-time,
and short-term computing. Therefore, cloud computing and edge computing can complement each other to match complex user demand scenarios and enlarge the application value of cloud computing and edge computing (Wu et al., 2019).

In reality, as users’ requirements are usually complex, a single service with limited functionality can not meet users’ expectations to use the services for complex tasks (Liu et al., 2016). Service composition combines multiple existing services in a specific logical order to complete complex tasks that a single service can not achieve. According to Gartner’s 2021 predictions, the number of services deployed in cloud and edge computing environments will grow explosively (Wang et al., 2017). A large number of services provide users with rich resources and bring new difficulties to users. One of the difficulties is how to select appropriate cloud and edge services among many candidate services to meet the complex needs of mobile users.

Usually, the above-mentioned service selection problem in the cloud and edge computing environment is affected by several factors: wireless data transmission speed, user movement, and connections between edges and clouds (Du et al., 2019). Selecting proper services out of candidate ones in such an environment can be modeled as a multi-objective optimization problem, which evolutionary algorithms can solve (Deb, 2014). This paper proposes an approach based on moth-flame optimization algorithm (MFO) and subjective and objective weighting methods to solve the service selection problem. The reason to use the Subjective and Objective weighting methods it effectively combines the subjective will of the user with the objective properties of the service. Recent studies indicate that the Moth-Flame optimization algorithm has a fast convergence speed and global search ability to produce competitive outputs in an unknown search space.

The main contributions of this paper are as follows:

1. This paper proposes a differential evolutionary moth-flame optimization algorithm (DEMF), which introduces the crossover and mutation operators of the differential evolution algorithm into the moth-flame optimization algorithm. In doing so, the early convergence and local extremum problems in the MFO could be solved, as shown by the comparative experimental results.
2. This paper proposes a subjective and objective weight calculation method to select services. Firstly, by using bias factors, users’ subjective requirements and the services’ objective performances are weighted respectively in terms of QoS attributes. Secondly, a combination method is proposed to synthesize the overall QoS values based on subjective and objective values of services. This method is used in the MFO to select a proper set of services iteratively to fulfill users’ needs.

In the rest of this paper, preliminary knowledge, the framework of proposed approach, data structure and algorithms, experimental results, related work and the conclusion are introduced and provided.

PRELIMINARY

This section introduces the necessary knowledge of service, mobile path, mobile edge calculation formula, and the moth-flame optimization algorithm.

Service and Quality of Service

A service \( w \) is represented as \((S, w_{\text{in}}, w_{\text{out}}, Q)\). \( S \) represents an edge server or a cloud server where the service is deployed, \( w_{\text{in}} \) denotes input parameters, \( w_{\text{out}} \) denotes the output parameters, and \( Q \) is a set of Quality of Services (QoS) attributes.

The service functionalities are described by \( w_{\text{in}} \) and \( w_{\text{out}} \), while the nonfunctionalities are described by QoS attributes (Li & Yan, 2016). In this paper, response time and cost are chosen to evaluate QoS of service, as users are usually sensitive to these two attributes. The service cost indicates the
price paid by a user for using the service, and the service’s response time means the time duration for a service to process inputs and generate outputs (Li et al., 2019). Usually, the cost and response time are negative attributes, which means that the higher the value, the lower the quality of service.

Considering that a user’s service request may be very complex and consists of a collection of tasks, selecting a proper set of services to complete the tasks of the request collaboratively becomes a challenge. Especially in the cloud and edge computing environment, services with similar functionalities on different servers and user mobility bring extra difficulties.

A composition of two services can accomplish a more complex request than a single service can (Zhu et al., 2019). Given two service \( w_i \) and \( w_j \), there are three kinds of structure for composing services, which are sequential \( (w_i; w_j) \), concurrent \( (w_i || w_j) \), and selective \( (w_i \text{ or } w_j) \) structures (Li et al., 2018). For each structure, the QoS values in terms of the cost and response time of service composition in the cloud and edge computing environment are calculated as follows:

The cost \( C \) of two connected services \( (w_i, w_j) \) is calculated as follows:

\[
C(w_i; w_j) = \sum \left\{ C(w_i), C(w_j) \right\}
\]  

(1)

\[
C(w_i || w_j) = \sum \left\{ C(w_i), C(w_j) \right\}
\]  

(2)

\[
C(w_i \text{ or } w_j) = \min \left\{ C(w_i), C(w_j) \right\}
\]  

(3)

The time delay \( R \) for connecting two services \( w_i \) and \( w_j \) is calculated as follows, given that \( w_i \) can be deployed on server \( s_k \) \((w_i \in s_k)\) and \( w_j \) can be deployed on server \( s_p \) \((w_j \in s_p)\):

\[
R(w_i; w_j) = \begin{cases} 
R(w_i) + R(w_j), & \text{if } w_i \in s_k, w_j \in s_k \\
R(w_i) + R(w_j) + R(s_p), & \text{if } w_i \in s_k, w_j \in s_p 
\end{cases}
\]  

(4)

\[
R(w_i || w_j) = \begin{cases} 
\max \left\{ R(w_i), R(w_j) \right\}, & \text{if } w_i \in s_k, w_j \in s_k \\
\max \left\{ R(w_i) + R(s_k), R(w_j) + R(s_p) \right\}, & \text{if } w_i \in s_k, w_j \in s_p 
\end{cases}
\]  

(5)

\[
R(w_i \text{ or } w_j) = \begin{cases} 
\min \left\{ R(w_i), R(w_j) \right\}, & \text{if } w_i \in s_k, w_j \in s_k \\
\min \left\{ R(w_i) + R(s_k), R(w_j) + R(s_p) \right\}, & \text{if } w_i \in s_k, w_j \in s_p 
\end{cases}
\]  

(6)

As different QoS attributes are measured in different scales and units, the QoS values of various attributes need to be normalized in comparison (Zhu et al., 2019). In this paper, to compare cost and response time, Equation (7) is used to normalize the values of these two attributes:
Model of Mobile Path

In the cloud and edge computing environment, the mobility of a user usually affects data transmission of uploading and downloading between the user and edge servers, which has significant impacts on the selection of proper services. In this paper, the mobile path of a user is studied by following a random waypoint model (Bettstetter et al., 2002). The continuous moving path is modeled as a sequence of segments. In each segment, the authors assume that the network connection condition between an edge/cloud server and the user remains stable, and the user’s moving direction and speed are kept constant.

With these assumptions, Figure 1 illustrates a user’s moving path segment, and the user’s mobile device connects to an edge server. Initially, the distance between the user and the edge server is $D$. Then, the user moves with an unchanged direction angle $\theta$, and the total length of the movement (the segment) depends on the speed $v$ of the user and the time $t$ the user has traveled. At the end of the segment, the distance $d$ between the user and the edge server is calculated as follows (Bettstetter et al., 2002):

$$
d = \sqrt{D^2 - 2 \times D \times v \times t \times \cos \theta + \left(v \times t\right)^2}$$

(Figure 1. Mobile Path)
Subjective and Objective Weights of QoS Attributes

The subjective weight of QoS attributes of service reflects a user’s preference (Lu & Yuan 2018). The heavier the subjective weight of the attribute, the greater the impact of the attribute on the user. Unlike the subjective weight, the objective weight of QoS attributes is based on the intrinsic characteristics of the attributes. Combining subjective and objective methods may integrate the merits of both objective features of the QoS attribute of services and users’ subjective preferences on services attributes together (Hongjiu & Yanrong, 2015). This paper exploits the order relation analysis method (Zhongxun et al., 2016), to synthesize subjective weights of QoS attributes. Compared with the widely used Delphi (Kim et al., 2013) and AHP (Analytic Hierarchy Process) (Liu et al., 2020) methods, the order relation analysis method does not require experts to repeatedly participate when obtaining weight indicators and adjusting the judgment matrix (Qiao et al., 2014). Besides, the algorithm’s complexity is significantly reduced while the efficiency is improved.

To calculate the subjective weights of QoS attributes, a user needs to compare the attributes pair wisely and specify the preferences at first. Supposing $q_i$ (resp. $q_{i-1}$) represents the $i$th (resp. $i$-1th) QoS attribute of service, the importance relationship between $q_i$ and $q_{i-1}$ is denoted by $r_i = \frac{q_{i-1}}{q_i}$, where the available values of $r_i$ is specified in Table 1.

The subjective weight $w^i_{\text{sub}}$ of the $i$th QoS attribute in the ordinal relationship can be calculated according to Equations (9) and (10).

$$W^i_{\text{sub}} = \left(1 + \sum_{m=2}^{k} \prod_{l=m-1}^{k} r_l\right)^{-1}$$ (9)

$$W^{i-1}_{\text{sub}} = r_i W^i_{\text{sub}}$$ (10)

where $k$ is the number of QoS attributes.

| Value of $r_i$ | Relationship | Explanation |
|----------------|--------------|-------------|
| 1.8            | Extremely important | $q_{i-1}$ is Extremely important than $q_i$ |
| 1.6            | Strongly important | $q_{i-1}$ is Strongly important than $q_i$ |
| 1.4            | Obviously important | $q_{i-1}$ is Obviously important than $q_i$ |
| 1.2            | Slightly important | $q_{i-1}$ is Slightly important than $q_i$ |
| 1.0            | Equally important | $q_{i-1}$ is Equally important than $q_i$ |

To calculate the objective weights of QoS attributes, the standard deviation method (Majumder, 2015), deviation maximization method (Şahin & Liu, 2016), coefficient of variation method (Eldridge et al., 2006) and entropy weight method (Rao & Tekabu, 2018) are exploited. In this paper, the coefficient of variation method is used for objective weight analysis. The coefficient of variation method can accurately reflect the dispersion between QoS attribute values. The larger the coefficient of variation, the more discrete the QoS attribute value, and vice versa. Given a set of tasks, each of
which can be completed by a set of services, and the number of QoS attributes of each service is $k$, below The authors show how to calculate the objective weight of an attribute with the coefficient variation method.

The average value of $i$th QoS attribute of services, $\bar{x}$, is calculated in Equation (11).

$$\bar{x} = \frac{1}{m} \sum_{j=1}^{m} x_{ij}$$  \hspace{1cm} (11)

where $m$ is the number of candidate services, $x_{ij}$ is the value of the $i$th QoS attribute of the $j$th service in the candidate services.

The mean square deviation of values of the $i$th QoS attribute, $s_i$, is calculated in Equation (12).

$$s_i = \sqrt{\frac{1}{m} \sum_{j=1}^{m} (x_{ij} - \bar{x})^2}$$  \hspace{1cm} (12)

The variation coefficient of values of the $i$th QoS attribute, $c_i$, is calculated in Equation (13).

$$c_i = \frac{s_i}{\bar{x}}$$  \hspace{1cm} (13)

The objective weight of values of the $i$th QoS attribute, $W_{ob}^i$, is calculated in Equation (14).

$$W_{ob}^i = \frac{c_i}{\sum_{1}^{k} c_i}$$  \hspace{1cm} (14)

where $k$ is the total number of QoS attributes of a service.

**Moth-Flame Optimization Algorithm**

The MFO has received wide attention since it was proposed in 2015 because of its strong optimization-seeking and fast convergence capabilities (Mirjalili, 2015). The MFO is inspired by extrapolation from the special navigation mechanism of natural moths flying spirally around flames. A moth is represented as the starting position of a spiral structure around a flame. A flame is represented as the ending position of the spiral structure. A position of moth or flame is a solution to the problem in the search space. Usually, a set of moths fly towards a set of flames to find the optimal solution to the problem iteratively. Flames are the best positions of moths obtained so far in each iteration. After each iteration, flames update positions by selecting the best positions obtained currently, and moths update positions accordingly to search around the updated flames. Specifically, a moth can update its position $M_i$ according to Equation (15):

$$M_i = D_i \times e^{bt} \times \cos 2\pi t + F_j$$  \hspace{1cm} (15)
where $D_i$ denotes the distance between the $i$th moth and the $j$th flame; $b$ is the predefined logarithmic spiral shape constant as the path coefficient, $t$ is a random number in $[-1,1]$; $F_j$ represents the position of $j$th flame. The expression of $D_i$ is as follows:

$$D_i = |F_j - M_i|$$ (16)

where, $M_i$ represents the current position of $i$th moth; $F_j$ represents the position of $j$th flame; $D_i$ denotes the distance between the $i$th moth and the $j$th flame.

To balance the global search capability and the local exploitation capability, in the MFO, the number of flames $f$ is reduced adaptively according to Equation (17), when the number of iterations increases.

$$f = round\left(M - l \times \frac{N - 1}{T}\right)$$ (17)

where $l$ is the current number of iterations, $N$ is the maximum number of flames, $T$ denotes the maximum number of iterations, and $M$ is the maximum number of moths. To illustrate the process of the MFO, the flow chart of the algorithm is shown in Figure 2.

**Figure 2. The Flow Chart of the MFO**
THE ARCHITECTURE AND ALGORITHMS

This section mainly introduces the architecture of the proposed approach, the DEMF, and the fitness function for selecting services by the DEMF.

Architecture

The architecture of the proposed approach is illustrated in Figure.3, which consists of three modules: the cloud and edge environment module, the service selection module, and the movement path module.

Cloud and edge environment: This module describes the distribution of services over the network of cloud and edge servers, specifies the connection information between edge servers, cloud servers and mobile users, and captures data transfer information between the edge servers, cloud servers and the users.

Service selection: This module selects proper services from a number of candidate services to fulfill the user’s complex request. Firstly, for each QoS attribute, an attribute fitness function is designed to calculate the overall QoS value of selected services based on the subjective and objective weights, along with considering the user’s movement and the structure of services composition. Secondly, a synthesis fitness function is proposed to calculate the synthesized value of all QoS attributes of selected services based on the results of all attribute fitness functions, and the DEMF algorithm uses the synthesized fitness function to select optimal services iteratively.

Movement path: This module records the user’s movement positions, moving directions, and speeds.

Besides, in this paper, the authors make the following assumptions:
1. All edge servers and cloud servers set up in the environment are interconnected.
2. The time delay of data transmission between a mobile user and a cloud server is greater than that between a mobile user and an edge server.
3. Always, there is more than one candidate service deployed on cloud and edge servers to complete a task proposed by a user.

The Attribute Fitness Functions Based on QoS Attributes

For each QoS attribute, an attribute fitness function is designed based on the subjective and objective weights of the selected services, and the results from all fitness functions are synthesized as the overall QoS value to evaluate the quality of selected services. In this paper, response time and cost are the selected QoS attributes. The lower the cost or response time, the higher the fitness value.

To calculate the uploading time and downloading time, the transmission time delay between the user and the edge server needs to be considered, which can be calculated by Equation (18).

\[
T(e) = \frac{\sum \text{Size}_{\text{task}}}{B \times \log_2 \left(1 + \frac{t_p \times g}{\sigma} \right)}
\]

where \(\text{Size}_{\text{task}}\) denotes the size of the data uploaded or downloaded, \(t_p\) is the wireless transmit power of the mobile device according to the Shannon-Hartley formula, \(B\) is bandwidth and \(\sigma\) is the noise at the receiver. \(g = d^{\alpha}\), \(d\) denotes the distance between the mobile device and the edge server, and \(\alpha = 4\) is the path loss factor.

Suppose that there are two edge servers \(e_i\) and \(e_j\), a cloud server \(c_k\), and a service \(s\). If the service \(s\) is deployed on edge server \(e_i\), in that case, the uploading time can be calculated by selecting the minimum time between uploading directly to edge server \(e_i\) and uploading to edge server \(e_j\) then to \(e_i\). If the service \(s\) is in cloud \(c_k\), the uploading time is the sum of the time to upload the request to the edge server and the time to send it from the edge server to cloud \(c_k\). The uploading time, \(T_{\text{up}}\), and downloading time, \(T_{\text{down}}\), can be calculated by Equations (19) and (20), respectively.

\[
T_{\text{up}}(\text{server}) = \begin{cases} 
\min \left\{ T(e_i), T(e_j) + T(e_j)(e_i) \right\}, & \text{if } s \in e_i \\
\min \left\{ T_{\text{up}}(e_i) + T(e_i)(c_k), T_{\text{up}}(e_j) + T(e_j)(c_k) \right\}, & \text{if } s \in c_k
\end{cases}
\]

\[
T_{\text{down}}(\text{server}) = \begin{cases} 
\min \left\{ T(e_i), T(e_j) + T(e_j)(e_i) \right\}, & \text{if } s \in e_i \\
\min \left\{ T_{\text{down}}(e_i) + T(e_i)(c_k), T_{\text{down}}(e_j) + T(c_k)(e_j) \right\}, & \text{if } s \in c_k
\end{cases}
\]

where \(T(e)\) represents the response time from the user to edge server \(e\), \(T(e)(e)\) represent the response time from cloud server \(c\) to edge server \(e\), and \(T(e_i)(e_j)\) represents the response time from edge server \(e_i\) to edge server \(e_j\).

The total time spent for a service selection solution, \(T_{\text{sum}}\), is the sum of the normalized request uploading time, the normalized execution time of the composed services, and the normalized solution downloading time in Equation (21).
\[ T_{\text{sum}} = \text{Norm}(T_{\text{up}}) + \text{Norm}(T_{\text{comp}}) + \text{Norm}(T_{\text{down}}) \] (21)

where \( T_{\text{up}} \) represents the uploading time of the request, \( T_{\text{down}} \) represents the downloading time of the solution, \( T_{\text{comp}} \) represents the execution time of all selected services to complete the tasks, which is based on the task composition structures and Equations (4-6).

The total cost of a service selection solution, \( C_{\text{sum}} \), is the sum of the normalized cost of the selected services in Equation (22).

\[ C_{\text{sum}} = \sum_{i=1}^{n} \text{Norm}(C_{s_i}) \] (22)

where a request has \( n \) tasks, each task has a service \( s_i \) that completes it, and \( C_{s_i} \) is the cost for the service \( s_i \).

**The Overall QoS Value**

The overall QoS value of selected services is calculated by combining the objective and subjective weights of attributes, which is used to analyze the quality of selected services in this paper. The subjective weight is calculated by the ordinal relation method according to Equations (9) and (10). The objective weight is calculated using the coefficient of variation method, and the weight of each attribute of the candidate services is calculated by Equations (11)-(14).

After calculating the subjective weights and objective weights of all attributes of selected services, the combined value of subjective and objective weights of an attribute, \( w_{\text{comb}}^{i} \), can be calculated by Equation (23).

\[ W_{\text{comb}}^{i} = \frac{w_{\text{sub}}^{i} \times w_{\text{ob}}^{i}}{\sum_{i=1}^{k} w_{\text{sub}}^{i} \times w_{\text{ob}}^{i}} \left\{ \alpha W_{\text{sub}}^{i} + (1 - \alpha) W_{\text{ob}}^{i} \right\} \] (23)

where \( k \) is the total number of QoS attributes, \( \alpha \) is a bias factor to adjust the subjective and objective weights, \( w_{\text{sub}}^{i} \) is the subjective weight of the \( i \)th QoS attribute, \( w_{\text{ob}}^{i} \) is the objective weight of the \( i \)th QoS attribute, \( w_{\text{comb}}^{i} \) is the overall QoS value of the \( i \)th QoS attribute.

In this paper, the cost and response time are the selected QoS attributes to be analyzed, the overall QoS value representing the quality of selected services is calculated as follows:

\[ \text{OverallQoS} = C_{\text{sum}} \times W_{\text{comb}}^{\text{cost}} + T_{\text{sum}} \times W_{\text{comb}}^{\text{time}} \] (24)
Differential Evolutionary Moth-Flame Optimization Algorithm

Each moth/flame in the Moth-Flame Algorithm represents a possible solution to the problem. The moth-flame algorithm is widely used due to its high parallel optimization capability, excellent globalization, and fast convergence (Mirjalili, 2015). However, the MFO suffers from premature convergence problems and easily falls into local extremes (Kaur et al., 2020). Therefore, the authors propose an optimization strategy for the MFO by combining variation and crossover from differential evolution with the MFO to solve the early convergence and local extremum problems. The optimization strategy is explained in Algorithm 1.

The algorithm is shown in Figure 4. The position of each moth represents a solution of service selection, and the flame represents the current optimal solution. By changing the position vector, the moths can fly in a high-dimensional space. In the service selection, \( n \) denotes the number of tasks, \((s_1, s_2, \ldots, s_n)\), each task contains \( m \) candidate services. Service \( w_1 \) is selected for \( s_1 \), service \( w_2 \) is selected for \( s_2 \), and service \( w_n \) is selected for \( s_n \). All selected services constitute solution \( F(w_1, w_2, \ldots, w_n) \).

In this algorithm, if it is the first iteration, the number of moths is equal to the number of flames (lines 6-8). Flames with poor benefit values are eliminated by the flame elimination formula (line 9). Then, moths update positions based on new benefit values (lines 11-15). The positions of the flames are updated (lines 17-19). Based on the differential evolution algorithm, this algorithm uses
the selects three randomly selected positions in the population by using the scaling factor \((F)\) and the crossover probability \((CR)\), then generates the variance vector (lines 21-24) and takes the better ones to the next generation. During iterations, the moth positions are updated by this method to enhance the dynamics of moth movement, maintain the diversity of positions within the search range, and get rid of the plague of getting into local extremes. \(T\) denotes the maximum number of iterations, \(M\) is the maximum number of moths and flames, and the algorithm’s time complexity is \(O(TM^2)\).

**CASE STUDY**

In this section, the authors present a meaningful example to calculate the overall QoS value that the DEMF can use to select optimal services. The authors assume that a user can send requests to edge servers when the user is within the wireless transmission range of the edge servers, the user can connect to all cloud servers, and candidate services are randomly assigned to the cloud or edge servers. As shown in figure 5, there are two edge servers \(e_1\) and \(e_2\), one cloud server \(c\), and a mobile user. The user sends a request with three tasks \(Task_1\), \(Task_2\), and \(Task_3\), which are executed sequentially. Each task has three candidate services.

**Figure 5. Case Study**

![Figure 5. Case Study](image)

Table 2 shows the relationship between tasks and services, and the response time (\(#\text{Res}\) ) and cost (\(#\text{Cost}\) ) are selected as QoS attributes. For example, \(Task_1\) has three candidate services \(srv_1^1\), \(srv_1^2\), and \(srv_1^3\).
Table 2. Tasks and Candidate Services.

| Task | Service(#Res/#Cost) | Service(#Res/#Cost) | Service(#Res/#Cost) |
|------|---------------------|---------------------|---------------------|
| Task 1 | srv_{1}^{1} (52/12) | srv_{1}^{2} (30/60) | srv_{1}^{3} (89/51) |
| Task 2 | srv_{2}^{1} (87/84) | srv_{2}^{2} (57/94) | srv_{2}^{3} (32/11) |
| Task 3 | srv_{3}^{1} (58/84) | srv_{3}^{2} (51/99) | srv_{3}^{3} (53/29) |

Table 3 shows the distribution of services on the edges and cloud. For example, srv_{1}^{1}, srv_{1}^{2}, and srv_{1}^{3} are deployed on edge e_{1}.

Table 3. Service Distribution on the Edges and Cloud.

| Server | Service | Service | Service |
|--------|---------|---------|---------|
| e_{1}  | srv_{1}^{1} | srv_{1}^{2} | srv_{1}^{3} |
| e_{2}  | srv_{2}^{1} | srv_{2}^{2} | srv_{2}^{3} |
| c      | srv_{3}^{1} | srv_{3}^{2} | srv_{3}^{3} |

Table 4 provides the setting of parameters used in the mobile cloud and edge environment. The authors assume the transmission delay between edge servers is 0.5ms, and the transmission delay between the edge server and the cloud server is 10ms. The initial distance between the user and e_{1} is d_{1}=100m with \( \theta_{1}=60° \), the initial distance between the user and e_{2} is d_{2}=200m with \( \theta_{2}=30° \). The data size of the uploading request is 1000M, and the data size of downloading request is 100M. The wireless transmission power is t_{p}=100W, the bandwidth is B=100Mbps, and the noise at the receiver is \( \sigma=10^{-8}W \). The user’s movement speed is v=1.5m/s.

Table 4. Parameters of the Mobile Cloud and Edge Environment.

| Parameter                          | Value          | Parameter                          | Value          |
|------------------------------------|----------------|------------------------------------|----------------|
| d_{1}                              | 100m           | Uploading Request                  | 1000M          |
| d_{2}                              | 200m           | Downloading Result                 | 100M           |
| \( \theta_{1} \)                   | 60°            | \( \sigma \)                       | 10^{-8}W       |
| \( \theta_{2} \)                   | 30°            | t_{p}                              | 100W           |
| v                                  | 1.5m/s         | B                                  | 100Mbps        |
| Transmission Delay between Cloud and Edge | 10ms          | Transmission Delay between Edges   | 0.5ms          |
Specifically, the authors use a candidate service selection $srv_1^1$, $srv_2^3$, $srv_3^2$ to explain the process of calculating the synthesized overall QoS value, which can be used by the DEMF for service selection. Detailed steps are provided as follows:

**Step 1:** Calculate the request uploading time.

According to Equation (18) and Table 4, the transmission time of uploading the request from the user to $e_1$ directly is $T(e_1) = \frac{1000}{100 \times \log_2 \left(1 + 100 \times \frac{100^{-4}}{10^{-8}}\right)} = 1.5$ms, and the transmission time of uploading the request from the user to $e_2$ directly is $T(e_2) = \frac{1000}{100 \times \log_2 \left(1 + 100 \times \frac{200^{-4}}{10^{-8}}\right)} = 3.5$ms.

Because $srv_1^1$ is the first service that receives the request and is located on $e_1$, the shortest time of uploading the user’s request to $e_i$ is $T_{up}(e_i)$.

$$T_{up}(e_1) = \min \{T(e_1), T(e_2) + T(e_2)(e_1)\} = \min \{1.5, 3.5 + 0.5\} = 1.5$$

**Step 2:** Calculate the service composition time

The composition time of services $srv_1^1$, $srv_2^3$, $srv_3^2$ is $T_{comp}$, which can be calculated by Equation (4) with the sequential structure.

$$T_{comp} = R(srv_1^1) + R(e_1)(c) + R(srv_2^3) + R(c)R(e_1) + R(srv_3^2) = 52 + 10 + 32 + 10 + 51 = 155 \text{ms}$$

where $srv_1^1$ is located on $e_1$, $srv_2^3$ is located on $c$, $srv_3^2$ is located on $e_2$, the response time of each service is shown in Table 2, and the transmission delay between cloud and edge is 10ms.

So far, the time taken from uploading the request to completing the task is $T_{up}(e_1) + T_{comp} = 156.5$ms.

**Step 3:** Calculate the result downloading time

According to Equation (8), After 156.5ms, the distance between the user and $e_1$ is $d_3 = 99.88$m, and the distance between the user and $e_2$ is $d_4 = 199.79$m.

**Step 4:** Calculate the result downloading time

According to Equation (18) and Table 4, the transmission time of downloading the result from $e_1$ to the user directly is $T(e_1) = \frac{100}{100 \times \log_2 \left(1 + 100 \times \frac{99.88^{-4}}{10^{-8}}\right)} = 0.15$ms, and the transmission time of downloading the result from $e_2$ to the user directly is $T(e_2) = \frac{100}{100 \times \log_2 \left(1 + 100 \times \frac{199.79^{-4}}{10^{-8}}\right)} = 0.35$ms.
Because $srv_3^2$ is the last service that produces the result and is located on $e_2$, the shortest time of downloading the result from $e_2$ to the user is $T_{down}(e_2)$.

$$T_{down}(e_2) = \min \left\{ T(e_2), T(e_1) + T(e_1)(e_2) \right\} = \min \left\{ 0.35, 0.15 + 0.5 \right\} = 0.2$$

Step 5: Calculate the normalized QoS values

To synthesize the values of different QoS attributes, normalization of QoS values is required. According to Equation (7) and Table 2, each service’s response time and cost are normalized, as shown in Table 5. Specifically, in this example, $T(e_2) + T(e_1)(e_2) = 3.5+10 = 13.5$ms is considered as the maximal value and $T(e_1) = 1.5$ms is considered as the minimal value in terms of uploading time, and 99 is taken as the highest cost and 11 is taken as the lowest cost.

Table 5. services’ normalized QoS.

| Task   | Service1(#Res/#Cost) | Service2(#Res/#Cost) | Service3(#Res/#Cost) |
|--------|----------------------|----------------------|----------------------|
| $srv_1^1$ | 0.627/0.989         | $srv_1^2$ (1/0.443)   | $srv_1^3$ (0/0.545)   |
| $srv_2^2$ | 0.034/0.170         | $srv_2^2$ (0.542/0.057) | $srv_2^3$ (0.966/1)   |
| $srv_3^3$ | 0.5250.170          | $srv_3^2$ (0.644/0)   | $srv_3^3$ (0.610/0.795) |

Step 6: Calculate the total response time and cost of the service selection

According to Equation (21), the total time of a service selection $srv_1^1$, $srv_3^2$ and $srv_3^2$ is $T_{sum}$.

$$T_{sum} = \text{Norm}(T_{up}) + \text{Norm}(T_{comp}) + \text{Norm}(T_{down})$$
$$= 1 + 0.690 + 0.995 = 2.685$$

According to Equation (22), the total cost of the service selection $srv_1^1$, $srv_3^3$, $srv_3^2$ is $C_{sum}$.

$$C_{sum} = \text{Norm}(C) + \text{Norm}(C) + \text{Norm}(C) = 0.989 + 1 + 0 = 1.989$$

Step 7: Calculate the synthesized weights

1. Calculate subjective weights.

According to Table 1, the authors assume that the importanc parameter between response time and cost is strongly important. Based on Equations (9) and (10), the subjective weights of response time and cost are $W_{sub}^{time} = 0.615$ and $W_{sub}^{cost} = 0.385$ respectively.

2. Calculate objective weights
Based on Equations (11-14), the objective weights of response time and cost are $W_{\text{ob time}} = 0.379$ and $W_{\text{ob cost}} = 0.621$ respectively.

3. Calculate synthesized weights

Based on Equation (23), the synthesized weights for response time and cost are $W_{\text{comb time}} = 0.349$ and $W_{\text{comb cost}} = 0.357$ respectively, given the bias factor $\alpha=0.5$.

Step8: Calculate the overall QoS value

According to Equation (24), the overall QoS value is calculated below.

$$\text{OverallQoS}_{\text{T W C}} = T_{\text{sum}} \times W_{\text{comb time}} + C_{\text{sum}} \times W_{\text{comb cost}} = 2.685 \times 0.349 + 1.989 \times 0.357 = 1.647$$

Steps 1-8 show how to calculate the overall QoS value of a candidate service selection solution, which can be considered as the fitness value used by the DEMF to find the optimal service selection iteratively.

EXPERIMENT

To evaluate the performance of the proposed method, experiments are carried out in a simulated environment. In this section, the experimental settings are described first, and then experimental results are provided and analyzed.

Experimental Settings

In the experiments, the authors use the coordinates of base stations in the CBD of Melbourne (Lai et al, 2018) and assume that each base station is equipped with an edge server. Besides, the authors assume two cloud servers can provide services to users in the CBD, the transmission delay is 20-50ms between two cloud servers, 1-3ms between two interconnected edge servers, and 10-50ms between a cloud server and an edge server. Those cloud and edge servers constitute a simulated cloud and edge environment. In this environment, the authors assume there is a user who can submit tasks in a request to edges within wireless transmission coverage or clouds, while the user moves in the CBD and a sequence of segments represents the movement. For each task, the authors assume there are several candidate services to complete the task, and services are randomly distributed to edges and clouds. The user uploads the request (50MB). Once all tasks are finished, the user may download the result (1MB).

In this paper, two groups of datasets are used to perform the comparison. The first group of datasets contains dataset 1, dataset 2, and dataset 3 with 10, 20, and 30 tasks, respectively, and each task has 60 candidate services. The second group of datasets contains dataset 4, dataset 5, and dataset 6 with 30, 40, and 50 candidate services for each task, respectively, and each dataset has 20 tasks. The rationale behind using the two groups of datasets is that the performance of the DEMF can be analyzed by a different number of tasks with the same number of candidate services and by a different number of candidate services with the same number of tasks.

To perform the experiments, some parameters related to the user’s preference on QoS attributes, user’s movement, wireless transmission, and the DEMF are provided in Table 6.
Experimental Results

To analyze the performance of the proposed service selection approach, the authors compare the proposed DEMF with the widely-used artificial fish swarm algorithm (AFS) (Pourpanah et al., 2020), particle swarm optimization algorithm (PSO) (Bansal, 2019), ant colony optimization algorithm (ACO) (Akhtar, 2019) and the original moth-flame optimization algorithm (MFO) (Mirjalili, 2015) in selecting proper services that can satisfy user’s functional and QoS requirements.

Experimental results are shown in Table 7. It can be observed that the DEMF can find the solution with the highest fitness value with the minimal variance value in each dataset. This means that the DEMF can always find the service selection solution with the best overall QoS value compared with the AFS, PSO, ACO, and MFO.

Figure 6 provides the converging processes of the algorithms in different datasets. It is obvious that although the DEMF method may not always converge faster than other algorithms, it can always converge on the best fitness value. Specifically, compared with the original MFO, the DEMF can find the best result within fewer iterations than the MFO in 5 datasets. This is due to the modification to the MFO by using the crossover and mutation from the differential evolution algorithm, which lets the DEMF avoid falling into a local optimum that the MFO may experience.

### Table 6. Parameters.

| Parameter       | Value       | Parameter       | Value       |
|-----------------|-------------|-----------------|-------------|
| Uploading Request | 50MB        | $v_{\text{min}}$ | 1m/s        |
| Downloading Result | 1MB        | $v_{\text{max}}$ | 1.5m/s      |
| $\sigma$       | $10^4W$     | $d_{\text{min}}$ | 10m         |
| $I_{\text{p}}$ | 100W        | $d_{\text{max}}$ | 200m        |
| B               | 100Mbps     | Scaling Factor  | 2           |
| Cross Probability | 0.5        | QoS Important Parameter | 1.6        |

### Table 7. Experimental results.

| Group | Dataset (#Task/#Service) | Algorithm | Fitness Value | Variance |
|-------|--------------------------|-----------|---------------|----------|
|       | dataset 1 10/60          | ACO       | 2.12E+00      | 8.53E-02 |
|       |                          | PSO       | 2.11E+00      | 7.12E-02 |
|       |                          | AFS       | 2.21E+00      | 2.33E-02 |
|       |                          | MFO       | 2.41E+00      | 8.65E-02 |
|       |                          | DEMF      | 2.52E+00      | 1.40E-02 |
|       | dataset 2 20/60          | ACO       | 3.22E+00      | 1.07E-01 |
|       |                          | PSO       | 2.92E+00      | 7.22E-02 |
|       |                          | AFS       | 2.23E+00      | 2.02E-01 |
|       |                          | MFO       | 3.46E+00      | 6.86E-02 |
|       |                          | DEMF      | 3.68E+00      | 1.13E-02 |
|       | dataset 3 30/60          | ACO       | 2.88E+00      | 1.37E-01 |
|       |                          | PSO       | 3.07E+00      | 9.93E-02 |
|       |                          | AFS       | 3.32E+00      | 1.22E-01 |
|       |                          | MFO       | 3.46E+00      | 4.15E-02 |
|       |                          | DEMF      | 3.63E+00      | 1.46E-02 |

Table 7 continued on next page
As a promising computing model, mobile edge computing has received widespread attention in recent years. Especially in the mobile edge environment, extensive research has been conducted on issues such as user task assignment, task offloading, and program placement.

Wu et al. (2021) proposed a decentralized reactive architecture for the online edge user allocation problem in mobile edge computing environments, and a decentralised reactive method to generate real-time allocation decisions through a fuzzy control mechanism. Peng et al. (2021) propose a decentralized approach: DoSRA for the online edge IoT task scheduling and resource allocation problem. This method can reduce the average weighted unloading response time by up to more than 1/3 compared to traditional calculation methods. Abbas et al. (2017) made a comprehensive investigation on the relevant research and technical development in the field of MEC. It provides the definition, advantages, architecture, and application fields of MEC. Finally, the relevant research and future direction are emphasized. The authors of Mao (Mao et al., 2017) focused on seamlessly merging the two disciplines of wireless communications.

**RELATED WORK**

{| Group       | Dataset (#Task/#Service) | Algorithm | Fitness Value | Variance |
|-------------|--------------------------|-----------|---------------|----------|
|             | dataset 4 20/30          | ACO       | 2.75 E+00     | 4.66E-01 |
|             |                          | PSO       | 2.67 E+00     | 8.93E-02 |
|             |                          | AFS       | 2.89 E+00     | 9.28E-02 |
|             |                          | MFO       | 3.13 E+00     | 4.05E-02 |
|             |                          | DEMF      | 3.37 E+00     | 2.04E-02 |
|             | dataset 5 20/40          | ACO       | 3.04 E+00     | 8.02E-02 |
|             |                          | PSO       | 2.82 E+00     | 8.24E-02 |
|             |                          | AFS       | 2.65 E+00     | 1.81E-01 |
|             |                          | MFO       | 2.96 E+00     | 5.67E-02 |
|             |                          | DEMF      | 3.27 E+00     | 2.33E-02 |
|             | dataset 6 20/50          | ACO       | 2.91 E+00     | 5.34E-02 |
|             |                          | PSO       | 2.83 E+00     | 1.35E-01 |
|             |                          | AFS       | 3.03 E+00     | 8.34E-02 |
|             |                          | MFO       | 3.18 E+00     | 2.73E-02 |
|             |                          | DEMF      | 3.31 E+00     | 1.07E-02 |

Table 7 continued
and mobile computing. MEC system deployment, MEC mobility management, green MEC, and privacy-aware MEC were discussed. Advances in these directions will promote the transformation of MEC from theory to practice. Chen et al. (2015) designed a distributed computing offloading algorithm by studying the multi-user computing offload problem of mobile edge cloud computing in a multi-channel wireless interference environment, which can achieve Nash equilibrium. As the user scale increases, the algorithm can obtain a better calculation of unloading performance. Peng et al. (2019) regarded the problem of edge user allocation as online decision-making and evolvable process and proposed a mobile sensing and migration support method called MobMig to allocate users in real-time. Compared with the traditional method, this method achieved a higher user coverage and a lower redistribution rate.

Service selection, an important part of the service computing field, has been studied by a large number of scholars. For example, Deng et al. (2014) presented a correlation-aware service pruning (CASP) approach to service selection, which managed QoS correlation by accounting for all services that may be integrated into the best integrated and pruned services. An integrated learning approach for predicting QoS deficiencies in 5G network environments was proposed by Yin et al. (2017). Sun et al. (2019) proposed a cloud service selection with a standard interaction framework (CSSCI) that applied a fuzzy measure and Choquet integral to measure and aggregate non-linear relations between criteria. In addition, the article designed a priority-based CSSCI to solve service selection.
problems in the situation where there is a lack of historical information to determine criteria relations and weights. Wu et al. (2019) proposed a heuristic algorithm that combined genetic algorithm and simulated annealing algorithm to perform service selection in mobile edge computing systems to optimize the response time of service calls in mobile edge computing systems. This method greatly reduces the response time of service invocation in the mobile edge computing system.

CONCLUSION

For the decision problem between mobile users in edge computing and cloud computing, a subjective-objective weighting method is proposed to combine the subjective willingness of users with the attributes of the service itself in proportion. For the problems of the moth-flame optimization algorithm, the differential evolution algorithm is added to avoid falling into local maximum and premature convergence. The authors calculate the service response time and cost of composed services in the mobile edge environment and use an adaptation function to calculate the performance. Experimental results show that the proposed approach has a better computational performance compared with others.

In the future, the authors will consider more service attributes and increase the number of QoS attributes. In addition, base station mobility will be added and used to address the impact of user access traffic on the rate.

FUNDING AGENCY

This research was supported by the National Natural Science Foundation of China [62002205], the Taishan Scholars Program of Shandong Province [No.tsqn201909109], the Natural Science Excellent Youth Foundation of Shandong Province [ZR2021YQ45], and the Zibo Key R&D Plan (Zibo city and university integration platform construction development) [2020SNPT0055].

ACKNOWLEDGMENT

Jing Li is the corresponding author of this paper.
REFERENCES

Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2017). Mobile edge computing: A survey. IEEE Internet of Things Journal, 5(1), 450–465. doi:10.1109/JIOT.2017.2750180

Akhtar, A. (2019). Evolution of Ant Colony Optimization Algorithm--A Brief Literature Review. arXiv preprint arXiv:1908.08007.

Bansal, J. C. (2019). Particle swarm optimization. In Evolutionary and swarm intelligence algorithms (pp. 11–23). Springer.

Bettstetter, C., Hartenstein, H., & Perez-Costa, X. (2002, September). Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate. In Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems (pp. 7-14). doi:10.1145/570758.570761

Chen, X., Jiao, L., Li, W., & Fu, X. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, 24(5), 2795–2808. doi:10.1109/TNET.2015.2487344

Corcoran, P., & Datta, S. K. (2016). Mobile-edge computing and the Internet of Things for consumers: Extending cloud computing and services to the edge of the network. IEEE Consumer Electronics Magazine, 5(4), 73–74. doi:10.1109/MCE.2016.2590099

Deb, K. (2014). Multi-objective optimization. In Search methodologies (pp. 403–449). Springer. doi:10.1007/978-1-4614-6940-7_15

Deng, S., Wu, H., Hu, D., & Zhao, J. L. (2014). Service selection for composition with QoS correlations. IEEE Transactions on Services Computing, 9(2), 291–303. doi:10.1109/TSC.2014.2361138

Du, W., Lei, Q., He, Q., Liu, W., Chen, F., Pan, L., & Zhao, H. et al. (2019, July). Multiple energy harvesting devices enabled joint computation offloading and dynamic resource allocation for mobile-edge computing systems. In 2019 IEEE International Conference on Web Services (ICWS) (pp. 154-158). IEEE. doi:10.1109/ICWS.2019.00035

Eldridge, S. M., Ashby, D., & Kerry, S. (2006). Sample size for cluster randomized trials: Effect of coefficient of variation of cluster size and analysis method. International Journal of Epidemiology, 35(5), 1292–1300. doi:10.1093/ije/dyl129 PMID:16943232

Kaur, K., Singh, U., & Salgotra, R. (2020). An enhanced moth flame optimization. Neural Computing & Applications, 32(7), 2315–2349. doi:10.1007/s00521-018-3821-6

Kim, M., Jang, Y. C., & Lee, S. (2013). Application of Delphi-AHP methods to select the priorities of WEEE for recycling in a waste management decision-making tool. Journal of Environmental Management, 128, 941–948. doi:10.1016/j.jenvman.2013.06.049 PMID:23892135

Lai, P., He, Q., Abdelrazek, M., Chen, F., Hosking, J., Grundy, J., & Yang, Y. (2018, November). Optimal edge user allocation in edge computing with variable sized vector bin packing. In International Conference on Service-Oriented Computing (pp. 230-245). Springer. doi:10.1007/978-3-030-03596-9_15

Li, J., Fan, G., Zhu, M., & Yan, Y. (2019, July). Pre-joined semantic indexing graph for QoS-aware service composition. In 2019 IEEE International Conference on Web Services (ICWS) (pp. 116-120). IEEE. doi:10.1109/ICWS.2019.00029

Li, J., Yan, Y., & Lemire, D. (2016). Full solution indexing for top-k web service composition. IEEE Transactions on Services Computing, 11(3), 521–533. doi:10.1109/TSC.2016.2578924

Li, J., Yan, Y., & Zhu, M. (2018). A compressed graph representation for services composition. International Journal of Web and Grid Services, 14(3), 305–331. doi:10.1504/IJWGS.2018.092589

Liu, J., Hao, S., Zhang, X., Wang, C., Sun, J., Yu, H., & Li, Z. (2016, June). Research on web service dynamic composition based on execution dependency relationship. In 2016 IEEE world congress on services (SERVICES) (pp. 113-117). IEEE.

Liu, Y., Eckert, C. M., & Earl, C. (2020). A review of fuzzy AHP methods for decision-making with subjective judgements. Expert Systems with Applications, 161, 113738. doi:10.1016/j.eswa.2020.113738
Lu, L., & Yuan, Y. (2018). A novel TOPSIS evaluation scheme for cloud service trustworthiness combining objective and subjective aspects. *Journal of Systems and Software, 143*, 71–86. doi:10.1016/j.jss.2018.05.004

Majumder, A. (2015). Application of standard deviation method integrated PSO approach in optimization of manufacturing process parameters. In *Handbook of research on artificial intelligence techniques and algorithms* (pp. 536–563). IGI Global. doi:10.4018/978-1-4666-7258-1.ch017

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys and Tutorials, 19*(4), 2322–2358. doi:10.1109/COMST.2017.2745201

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249. doi:10.1016/j.knosys.2015.07.006

Peng, Q., Wu, C., Xia, Y., Ma, Y., Wang, X., & Jiang, N. (2021). *Dosra: A decentralized approach to online edge task scheduling and resource allocation*. IEEE Internet of Things Journal.

Peng, Q., Xia, Y., Feng, Z., Lee, J., Wu, C., Luo, X., & Chen, P. et al. (2019, July). Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In *2019 IEEE International Conference on Web Services (ICWS)* (pp. 91-98). IEEE. doi:10.1109/ICWS.2019.00026

Personè, V. D. N., & Grassi, V. (2019, July). Architectural issues for self-adaptive service migration management in mobile edge computing scenarios. In *2019 IEEE International Conference on Edge Computing (EDGE)* (pp. 27-29). IEEE. doi:10.1109/EDGE.2019.00020

Pourpanah, F., Wang, R., Lim, C. P., & Yazdani, D. (2020). *A review of the family of artificial fish swarm algorithms: Recent advances and applications*. arXiv preprint arXiv:2011.05700.

Qiao, C., Zhang, M., Chen, Y., & Wang, R. (2014). A quantitative judgement method for safety admittance of facilities in chemical industrial parks based on G1-variation coefficient method. *Procedia Engineering, 84*, 223–232. doi:10.1016/j.proeng.2014.10.429

Rao, D. K., & Tekabu, T. B. (2018, December). Calibrating design weights in stratified sampling using auxiliary information on coefficient of variation. In *2018 5th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)* (pp. 265-270). IEEE.

Şahin, R., & Liu, P. (2016). Maximizing deviation method for neutrosophic multiple attribute decision making with incomplete weight information. *Neural Computing & Applications, 27*(7), 2017–2029. doi:10.1007/s00521-015-1995-8

Sun, L., Dong, H., Hussain, O. K., Hussain, F. K., & Liu, A. X. (2019). A framework of cloud service selection with criteria interactions. *Future Generation Computer Systems, 94*, 749–764. doi:10.1016/j.future.2018.12.005

Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., & Wang, W. (2017). A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access: Practical Innovations, Open Solutions, 5*, 6757–6779. doi:10.1109/ACCESS.2017.2685434

Wu, C., Peng, Q., Xia, Y., Ma, Y., Zheng, W., Xie, H., Pang, S., Li, F., Fu, X., Li, X., & Liu, W. (2021). Online user allocation in mobile edge computing environments: A decentralized reactive approach. *Journal of Systems Architecture, 113*, 101904. doi:10.1016/j.sysarc.2020.101904

Wu, H., Deng, S., Li, W., Yin, J., Li, X., Feng, Z., & Zomaya, A. Y. (2019, July). Mobility-aware service selection in mobile edge computing systems. In *2019 IEEE International Conference on Web Services (ICWS)* (pp. 201-208). IEEE. doi:10.1109/ICWS.2019.00042

Yin, Y., Xu, Y., Xu, W., Gao, M., Yu, L., & Pei, Y. (2017). Collaborative service selection via ensemble learning in mixed mobile network environments. *Entropy (Basel, Switzerland), 19*(7), 358. doi:10.3390/e19070358

Zhu, M., Fan, G., Li, J., & Wang, F. (2019). A service composition approach based on overall QoS and modified graphplan. *International Journal of Web and Grid Services, 15*(4), 319–339. doi:10.1504/IJWGS.2019.103214
Ming Zhu is an Assistant Professor and the Assistant Dean at the School of Computer Science and Technology at Shandong University of Technology, Shandong, China. He received his PhD in Computer Science from the Concordia University. He has more than 20 publications in journals and conferences like IJWGS, SOCA, IJWET, ICWS, SCC, EUSPN, ANT, and etc. His research interests are related to service computing, process-oriented programming, event modeling, and concurrent systems.

Xiukun Yan is currently pursuing the M.S. degree in Computer Science and Technology, Shandong University of Technology. She is currently working on mobile edge computing. Her main research interests are edge computing and service computing.

Jing Li is an Associate Professor at the College of Computer Science and Technology at Shandong University of Technology, Shandong, China. She received her Doctoral degree in Computer Science at Concordia University, Montreal. She has more than 20 publications in journals and conferences like TSC, IJWGS, SOCA, ICWS, ICSOC, SCC, and etc. Her main research interests are service computing and database-based service composition. Jing Li is the corresponding author of this paper.

Cong Liu received the B.S. and M.S. degrees in computer software and theory from the Shandong University of Science and Technology, Qingdao, China, in 2013 and 2015, respectively, and the Ph.D. degree in computer science and information systems from the Eindhoven University of Technology, Eindhoven, the Netherlands, in 2019. He is currently a full professor in the School of Computer Science and Technology in Shandong University of Technology. He has more than 80 publications in journals and conferences like TSC, TSMC, TASE, TBD, TCC, TITS, IoTJ, FGCS, DSS, ICWS, ICPC, EuroVIS, and etc. His current research interests include business process management, process mining, and emergency management.

Yawen Cao is a student at Jinan Foreign Language School. Her research interests include service computing and artificial intelligence.