International Journal of Web Engineering and Technology

ISSN online: 1741-9212 - ISSN print: 1476-1289
https://www.inderscience.com/ijwet

Using seagull optimisation algorithm to select mobile service in cloud and edge computing environment

Feilong Yu, Jing Li, Ming Zhu, Xiukun Yan

DOI: 10.1504/IJWET.2022.10049995

Article History:

Received:	13 October 2021
Last revised:	03 May 2022
Accepted:	09 May 2022
Published online:	25 August 2022

Using seagull optimisation algorithm to select mobile service in cloud and edge computing environment

Feilong Yu, Jing Li, Ming Zhu* and Xiukun Yan

College of Computer Science and Technology,
Shandong University of Technology,
Zibo, China
Email: yfl_sdut@126.com
Email: li_jing@sdut.edu.cn
Email: zhu_ming@sdut.edu.cn
Email: yxksunshine@163.com
*Corresponding author

Abstract: With the rapid development of edge computing, more and more services are deployed on edge servers. Compared with traditional cloud computing, services in the edge computing environment are closer to users, which bring benefits of high performance and low latency to the user-service interactions. However, due to the limited resources of edges, services provided by edges alone may fail to meet increasingly complex mobile computing requirements; therefore, services on clouds become an effective supplement. With the massive increment of services in the mobile internet, selecting proper services to fulfil mobile users' requests becomes a key research field. This paper proposes a service selection model for mobile service selection problem in cloud and edge computing environment. The proposed model combines the seagull optimisation algorithm and the simulated annealing algorithm. Through comparative experiments on simulation datasets with referencing to some other service selection models, it can be inferred that the proposed selection model finds a solution with better QoS performance in fewer iterations.

Keywords: mobile edge computing; cloud computing; seagull optimisation algorithm; SOA; service selection.

Reference to this paper should be made as follows: Yu, F., Li, J., Zhu, M. and Yan, X. (2022) 'Using seagull optimisation algorithm to select mobile service in cloud and edge computing environment', *Int. J. Web Engineering and Technology*, Vol. 17, No. 1, pp.88–114.

Biographical notes: Feilong Yu is a graduate student at the Department of Computer Science and Technology at Shandong University of Technology, Shandong, China. His main research interests are edge computing and service computing.

Jing Li is an Associate Professor at the Department of Computer Science and Technology at Shandong University of Technology, Shandong, China. She received her doctoral degree in Computer Science at the Concordia University, Montreal. Her main research interests are service computing and database-based service composition.

Ming Zhu is an Assistant Professor at the Department of Computer Science and Technology at Shandong University of Technology, Shandong, China. He did his PhD in Computer Science at the Concordia University. His research interests are related to Service computing, process-oriented programming, event modelling, and concurrent systems.

Xiukun Yan is a graduate student at the Department of Computer Science and Technology at Shandong University of Technology, Shandong, China. Her main research interests are edge computing and service computing.

1 Introduction

Recently, the widespread popularity of intelligent mobile devices has resulted in huge requirements for cloud and edge computing. Applications, such as natural language processing, facial recognition, and video processing, in mobile devices are delay-sensitive and demand-intensive computations. Mobile edge computing optimises cloud computing by processing data at the edge of the network near the original data source (Huang et al., 2018). It is envisioned to help alleviate the resource scarcity of mobile devices. Therefore, users can gain easy access to lightweight services within shorter response time. Compared with edge computing, cloud computing can provide more computing resources (e.g., services). Collaborative work of cloud computing and edge computing has gradually become a promising trend in mobile internet development (Muhammad et al., 2020).

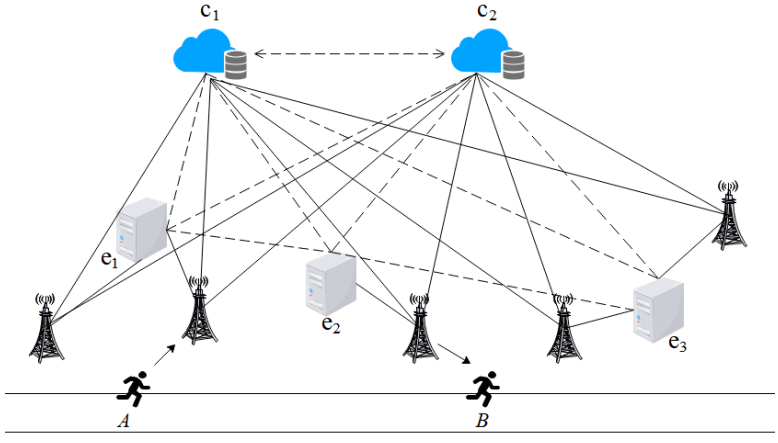
Figure 1 shows a typical scenario where a mobile user uses services in cloud and edge computing environment. There are three edge servers, two cloud servers, several base stations, and a mobile user in this example. These cloud and edge servers interconnect and able to transfer data with each other. The request submitted to a base station is further transmitted to an edge or cloud server. Assuming that the mobile user sends a request to the nearby base station at location A and receives a response message when the user moves to location B, the time interval is the sum of uploading time, the total response time of services and servers, and downloading time. The time of uploading/downloading, is decided by the size of the request and the transfer capacity (Wu et al., 2019).

However, challenges are still waiting to be solved in a mobile service selection problem in cloud and edge computing environment (Shi et al., 2016). First of all, users may experience long latency for data exchange with cloud resources. Besides, services provided by edge servers are limited and may fail to meet the user's requirement. Last but not least, users submit requests while moving, dynamic and ever-changing locations of users enhance difficulties of supplying stable services.

This paper addresses mobile service selection problem in edge and cloud computing environment with an extended seagull optimisation algorithm (ESOA) (Dhiman and Kumar, 2019). Inspired by the nature of seagulls' migration and attack behaviours, the seagull optimisation algorithm (SOA) is an evolutionary optimisation algorithm and is widely used in rolling bearing production and other aspects (Dhiman and Kumar, 2019). The SOA has high efficiency in solving unconstrained problems in unknown search space, has fast convergence speed, and is easy to implement. To solve service selection problem with the SOA, each candidate solution is represented as the location of a seagull

(see Section 2.6). The algorithm starts from a set of randomly generated locations and searches toward the location with either the minimal cost or response time.

Figure 1 A mobile user in edge and cloud computing environment (see online version for colours)



The specific contributions of this paper are summarised as follows:

We consider the mobile service selection problem in a cloud and edge computing environment, and formulate the problem to an optimisation problem.

- A mobile service selection model based on an ESOA is proposed. To the best of our knowledge, this is the first time the SOA combined with the simulated annealing algorithm has been applied to solve the mobile service selection problem.
- We evaluate the performance of the proposed algorithm through extensive experiments. Simulation results show that the performance of the proposed algorithm outperforms referring studies by at least 5.3% better response time in fewer iterations.

The rest of this paper is organised as follows: we introduce preliminary knowledge in Section 2. Section 3 discusses related work. Section 4 presents our proposed method and algorithms, and thereafter, a case study is given in Section 5 to explain how the proposed method is applied to solve a mobile service selection problem. Section 6 gives experimental results. Finally, the conclusion is drawn in Section 7.

2 Preliminary knowledge

In this section, the service selection problem, the model of mobile path, the SOA, and the simulated annealing algorithm are introduced.

2.1 Service and service combination

Definition 2.1: A service s is defined as (srv, s_{in}, s_{out}, Q). srv is the server where s is distributed and executed; s_{in} is the input data of the service. s_{out} is the output data of the service. Q is a finite number of non-functional properties of s.

Commonly, functional properties describe a service's input and output parameters, and non-functional properties are referred to as quality of service (QoS). Services with similar functionalities may have different QoS criteria. In this paper, we use cost and response time to measure QoS.

- Cost (C): the price paid by the user for using the service (unit: cent).
- Response time (R): the time interval between receiving a request and sending a result (unit: millisecond).

Given two services s_i and s_j, there are three kinds of service connection structures: sequential, concurrent, and selective. We use s_i; s_j to represent sequential connection, use $s_i \| s_j$ to represent concurrent connection, and use $s_i | s_j$ to represent selective connection.

The cost C of connected services (s_i and s_j) is calculated as follows:

$$C(s_i; s_j) = C(s_i) + C(s_j) \tag{1}$$

$$C(s_i \| s_j) = C(s_i) + C(s_j) \tag{2}$$

$$C(s_i | s_j) = \min\{C(s_i), C(s_j)\} \tag{3}$$

Response time R of connected services (s_i and s_j) is calculated as follows:

$$R(s_i; s_j) = \begin{cases} R(s_i) + R(s_j), \text{if } s_i \in srv_k, s_j \in srv_k \\ R(s_i) + R(s_j) + R(srv_k, srv_p), \text{if } s_i \in srv_k, s_j \in srv_p \end{cases} \tag{4}$$

$$R(s_i \| s_j) = \begin{cases} \max\{R(s_i), R(s_j)\}, \text{if } s_i \in srv_k, s_j \in srv_k \\ \max\{R(s_i) + R(srv_k, srv_x), R(s_j) + R(srv_p, srv_y)\}, \text{if } s_i \in srv_k, s_j \in srv_p \end{cases} \tag{5}$$

$$R(s_i | s_j) = \begin{cases} \min\{R(s_i), R(s_j)\}, \text{if } s_i \in srv_k, s_j \in srv_k \\ \min\{R(s_i) + R(srv_k, srv_x), R(s_j) + R(srv_p, srv_y)\}, \text{if } s_i \in srv_k, s_j \in srv_p \end{cases} \tag{6}$$

where srv_k and srv_p are servers, $s_i \in srv_k$ indicates srv_k hosts service s_i, and $R(srv_k, srv_p)$ means the data transmission time between srv_k and srvp. For equations (5)–(6), we assume that the successor service of s_i is located on server srvx and the successor service of s_j is located on server srv_y.

Definition 2.2: A service request contains the following components:

A finite set of tasks requested by the user, and each task can be fulfilled by a service:

- a combination structure of tasks
- the input data provided by the user.

To understand this paper, the notations used in the paper are presented in Table 1.

Table 1 Notations

Expression	*Description*	*Expression*	*Description*
srv	Server	s	Service
C	Cost	R	Response time
d	Distance	B	Bandwidth
T	Data transmission time	tp	Wireless transmit power
g	Channel power gain	h_0	Received power
σ^2	Noise at the receiver	γ	Path loss factor
v	Movement speed of user	e	Base of the natural logarithm
A	Movement behaviour of the seagull	BL	Variable to balance exploration and development
f_c	Conversion frequency	rd	A random variable of the SOA
x, y, z	Three planes in the space	h	Radius of each helix
α	Random angle	u, k	Correlation constants of spiral shape

2.2 Base station, edge server and cloud server

As modelled in Figure 1, each base station is connected to an edge server. An edge server may transfer data with several geographically closed base stations. Each edge server is connected to the Internet and is able to communicate data with other servers. Similarly, each cloud server is connected to the internet and is able to transfer data with other servers.

The network topology of base stations and servers is described by an incidence matrix, where each row represents a server, and each column represents a base station. Each cell in the matrix represents the data transmission speed between a server and a base station. If an edge server i is connected to a base station j, then the value of cell with index ij is the data transmission speed between them. Otherwise, the value is set to 0.

The network topology of edge and cloud servers is described by an adjacency matrix, in which each cell represents the data transmission speed between two servers. If two servers are connected, then the value of cell with index ij is the data transmission speed between them. Otherwise, the value is set to 0. We assume that the data transmission speed from a server to itself is maximised.

2.3 The model of mobile path

In the mobile edge and cloud computing environment, the mobile path of the user partially decides the time of uploading, service selection and downloading, and therefore influences the selection of services. Below, we model the user's mobile path.

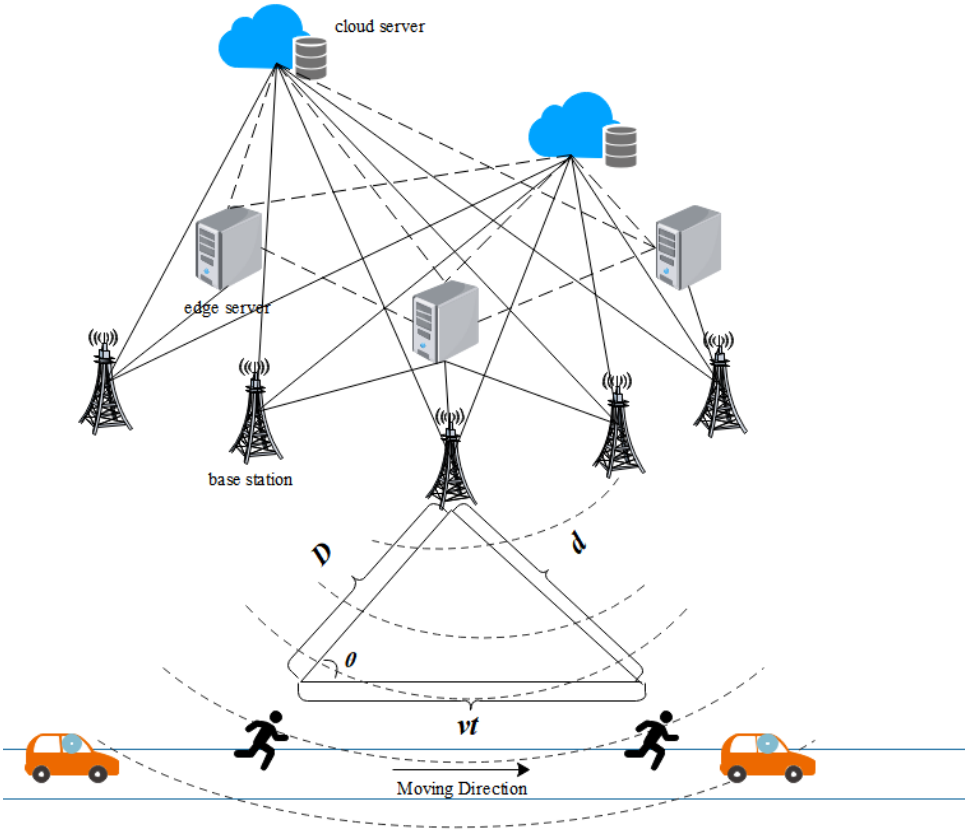
We use the random waypoint mobility model to simulate the mobile path (Bettstetter et al., 2004). The model in this paper is suitable for the scenario that the user's moving path is determined in advance. The mobile path is decomposed into a group of path segments, supposing the network condition, moving direction, and speed do not change in a path segment.

As shown in Figure 2, we assume that the initial distance between the user and the base station is D, the moving direction angle θ is unchanged, and the user moves at a constant speed v. After time t, the path segment can be calculated as vt, the distance d between the user and the base station is:

$$d = \sqrt{D^2 - 2Dvt\cos\theta + (vt)^2}. \tag{7}$$

If the user walks, v = 1.5 m/s; if the user rides a bike, v = 5 m/s; if the user take a campus bus, v = 10 m/s; if the user drives, v = 15 m/s.

Figure 2 The mobile path (see online version for colours)

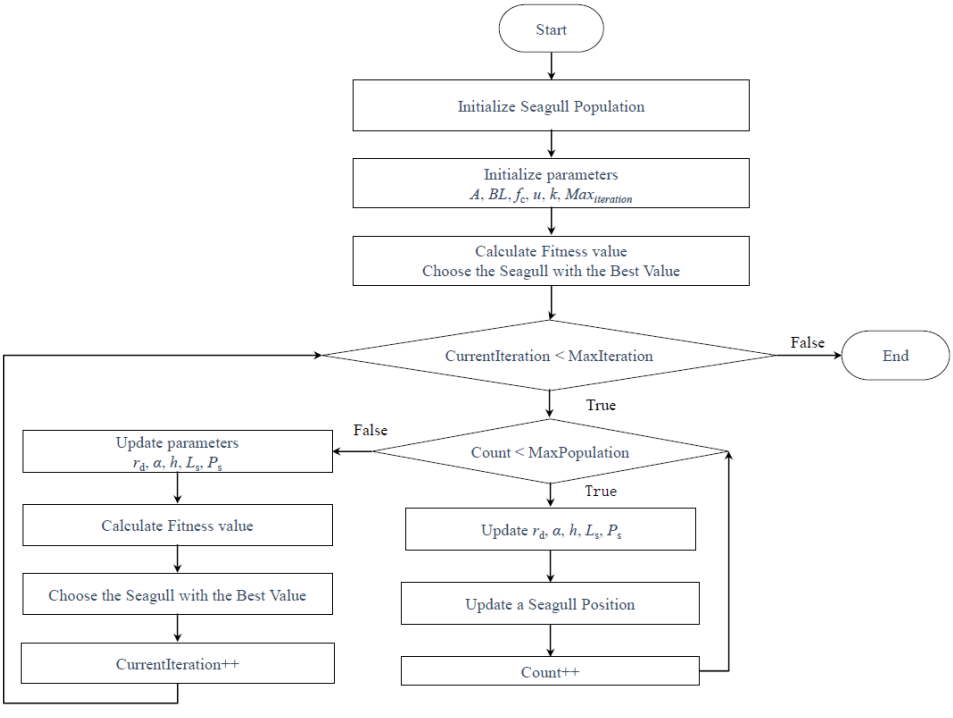


2.4 Seagull optimisation algorithm

The SOA is an evolutionary optimisation algorithm that imitates seagulls' migration and attack behaviours to find a near-optimal solution iteratively (DK19 and Kumar, 2019). The migration behaviour means seagulls fly to places that are more suitable for survival, which affects the algorithm's global exploration ability. The attack behaviour is the forage, which affects the algorithm's local development ability. Seagulls fly to new positions in each generation, and each position represents a candidate solution. The fitness value of each position is calculated, and seagulls with better positions are possible

to be brought to the next generation. After several generations, the algorithm gradually approaches the optimal solution. The flowchart of the SOA is shown in Figure 3.

Figure 3 Flowchart of SOA



2.4.1 Exploration ability

To avoid colliding with surrounding seagulls, the SOA uses a variable A to adjust the position $P_{ns}(t)$ of seagulls. A represents the movement behaviour of the seagull in a given search space. The new position $P_{ns}(t)$ of a seagull in t^{th} generation is calculated as follows:

$$P_{ns}(t) = A \times P_s(t) \tag{8}$$

where $P_s(t)$ is the current position of the seagull, and A is calculated as follows:

$$A = f_c - \left(t \times \left(f_c / Max_{iteration}\right)\right) \tag{9}$$

where f_c controls the conversion frequency of A from f_c to 0. In paper (Dhiman and Kumar, 2019), the value of f_c is analysed experimentally, and experimental results show that the algorithm achieves the optimal value when $f_c = 2$. $Max_{iteration}$ means the maximum generation.

$$M_s(t) = BL \times \left(P_{bs}(t) - P_s(t)\right) \tag{10}$$

$M_s(t)$ draws seagulls close to the best position $P_{bs}(t)$. BL balances exploration and development of the algorithm, the variation of BL follows:

$$BL = 2 \times A^2 \times rd \tag{11}$$

where rd is a random number that lies in the range of [0, 1]. The distance $L_s(t)$ between the seagull and the best seagull is calculated as:

$$L_s(t) = \left| P_{ns}(t) + M_s(t) \right| \tag{12}$$

2.4.2 Exploitation ability

In the process of exploitation, seagulls spiral descent and adjust their positions, attack angles, and speeds. As seagulls make spiral motion during attacking (Dhiman and Kumar, 2019). The location of a seagull is represented as a node in three dimensional coordinates:

$$x = h \times \cos(\alpha) \tag{13}$$

$$y = h \times \sin(\alpha) \tag{14}$$

$$z = h \times \alpha \tag{15}$$

$$h = u \times e^{\alpha k} \tag{16}$$

where h is the radius of each helix, α is a random angle in the range of 0 to 2π, u and k are correlation constants of spiral shape, and e is the base of the natural logarithm.

$$P_s(t) = L_s(t) \times x \times y \times z + P_{bs}(t) \tag{17}$$

The SOA starts with a randomly generated population, and seagulls constantly update their positions to find the optimal position. These features inspire us to employ the SOA algorithm to solve the mobile service selection problem in a cloud and edge environment.

2.5 Simulated annealing algorithm

The basic idea of the simulated annealing algorithm comes from the physical annealing process. When an object is heated to a completely high temperature, the particles in the solid become disordered and the energy increases significantly when the temperature rises. When the particles cool slowly, the particles gradually become orderly and the energy minimises. The whole cooling process is called annealing.

The simulated annealing algorithm starts from a randomly initial solution P_{sa} and a control parameter T_{sa}. A new solution P_{new} is generated as follows:

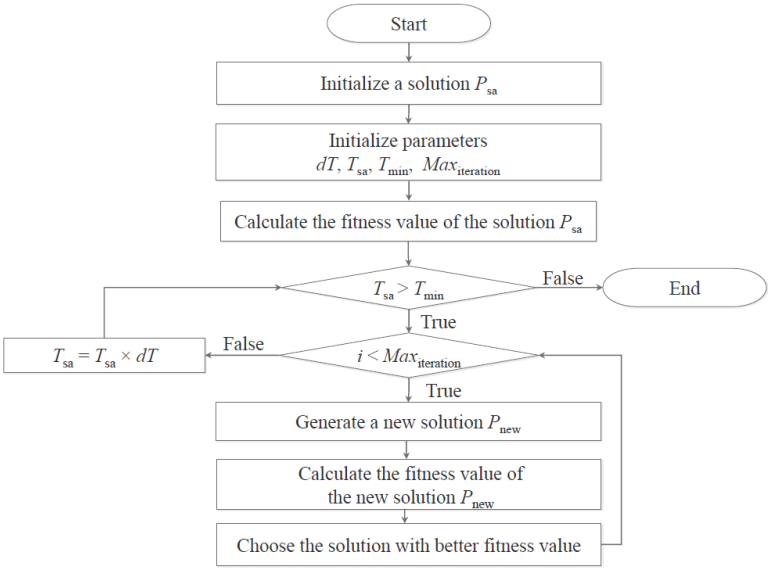
$$P_{new} = P_{sa} + \left(2 \times f - r_{\max}\right) \times T_{sa} \tag{18}$$

where f is a random number in the range of [0, $r_{\max}$]. T_{sa} is a temperature parameter controlled by the cooling rate dT in the range of [0, 1].

According to the fitness value of the new solution P_{new}, the algorithm chooses to accept or give up P_{new}, and T_{sa} gradually attenuates. Finally, an approximate optimal solution is returned.

This paper simplifies the simulated annealing algorithm and combines it with the SOA. The basic idea of the simulated annealing algorithm used in this paper is depicted in Figure 4.

Figure 4 Flowchart of simulated annealing algorithm



2.6 Mapping service selection problem with the ESOA algorithm

Given a service request, the mobile service selection problem in the edge and cloud computing environment is finding proper services to complete tasks while considering QoS requirements. The service selection problem consists of the following components:

- a service request
- a model of user's mobile path represented by a sequence of path segments
- a set of candidate services, where each task is fulfilled by a service
- a network of base stations, edge and cloud servers.

The goal of the mobile service selection problem is to find a combination of services to complete tasks with the minimal overall response time or cost, a base station receives user's request as input data, sends the request to an edge or a cloud server, after service combination, a solution is delivered to the user. This selection problem is an optimisation problem.

In order to solve a mobile service selection problem, we use the ESOA to find a near optimal solution, that is, to find an approximate optimal position of seagull. A seagull's position is represented as a combination of candidate services, one for each task. For a service request with n tasks (t_1, t_2, ..t_i .., t_n), a seagull position is represented as (s_1, s_2, ..s_j .., s_n), which means that service s1 is selected for task t_1, s_j is selected for t_i, and s_n is selected for t_n respectively. The service combination solution with the minimum overall response time or cost is selected and returned.

Following the SOA in Section 2.4, the initial seagull population and their corresponding positions are randomly generated. Fitness functions are designed (Section 4.2) to evaluate the position of each seagull, and the seagull with the best

position (fitness value) is selected. Then, a new seagull generation is produced by updating the position of each seagull in the previous generation. For each seagull, its new position is compared with the one in the previous generation, and the better one is retained. The seagull with the best position (fitness) is chosen as the current best seagull. The best seagull gets close to the optimal position without collision with other seagulls. This process ends when a satisfying position is obtained or reaches the maximum generation. The seagull with the best position (fitness) is returned to the user as the solution of the problem.

3 Related work

With the continuous development of mobile Internet technology, cloud and edge computing is widely applied in various fields. This part introduces the mobile service selection in cloud and edge and the evolutionary algorithms used in service selection.

3.1 Mobile service selection in cloud and edge computing

Service selection has been paid more and more attention by experts and scholars. Classic service selection is a problem that parses functional descriptions and matches services at the technical level (Reiff-Marganiec and Tilly, 2008). Due to the increasing number of services with similar functional properties on the web and to maximise users' satisfaction, researchers pay attention to non-functional (QoS) attributes. Gelenbe et al. proposed a method from the perspective of QoS and energy (Gelenbe and Lent, 2013). They regarded the choice between local services and cloud services as an optimisation problem and presented a mathematical model. Hentschel et al. presented a cloud brokering system that selects cloud services with matchmaking and ranking (Raoul et al., 2020). The proposed virtual broker as a service framework (ViBROS) is less costly for small and medium enterprises when choosing cloud services. In addition, it can be easily used by less-skilled users.

Orsini et al. (2016) proposed a cloud-aware adaptive middleware for mobile edge computing. This approach aims to link specific mobile cloud and edge computing requirements while employing combinatorial adaptation and sensor-based reasoning to provide dynamic adaptation to the configuration-free programming model. Huang et al. (2018) presented a simulation-based approach for QoS-aware service selection in a mobile edge computing environment. The optimisation goal can be softened by sequential optimisation to resolve the problem in an acceptable amount of time. Xu et al. (2019) proposed a computation offloading method (COM), to reduce the influence of offloading of mobile application. Specifically, they investigated a system model that computes mobile devices' execution time and energy consumption. Then dynamic schedules of data/control constrained computing tasks are confirmed. Deng et al. (2020) divided edge intelligence into AI for edge (intelligence-enabled edge computing) and AI on edge (artificial intelligence on edge). The former focuses on providing better solutions to key problems in edge computing with the help of popular and effective AI technologies, while the latter studies how to carry out the entire process of building AI models. Mach and Becvar (2017) classified research on computation offloading into three key areas: decision on computation offloading; allocation of computing resources within the MEC, and mobility management to improve the efficiency of computation offloading.

3.2 Cloud computing and edge computing

The combination of internet of things (IoT) and cloud computing enhance the capability of IoT, but it also brings corresponding disadvantages. In order to effectively solve internal attacks and improve the efficiency of IoT-cloud services, Wang et al. (2019) proposed a novel architecture that integrates a trust evaluation mechanism and service template with a balance dynamics based on cloud and edge computing, which improves the security and efficiency of IoT-cloud system. Ma et al. (2017) proposed a cloud assisted mobile edge computing (CAME) framework to enhance the computing capacity of the system, and used an optimisation problem to minimise the system delay and cost. This framework guarantees the lowest cost. The optimal resource provisioning (ORP) algorithm optimises the computation capacity of edge hosts and dynamically adjusts the cloud tenancy strategy (Ma et al., 2021).

Li and Wang (2018) studied edge server placement in the cloud and edge computing, and designed a particle swarm optimisation based energy-aware edge server placement algorithm. Zhang et al. (2017) developed a joint cloud and wireless resource allocation algorithm based on evolutionary game (JRA-EG) to solve the cost problem of mobile devices in a mobile edge computing environment, and obtained evolutionary equilibrium (EE) by replicate dynamics method. Because users' mobility leads to frequent switching of edge servers and increases the delay of task processing, Wang et al. (2021) studied the microservice coordination among edge clouds and proposed a dynamic programming-based offline microservice coordination algorithm. Hong et al. reviewed publications published from 2013 to 2018, detailed the architecture, infrastructure, and algorithms that support resource management in fog/edge computing, and described characteristics of the computing paradigm for handling user data (Hong and Varghese, 2019).

3.3 Evolutionary algorithms in service selection

Dahan et al. (2019) proposed an enhanced artificial bee colony (ABC) algorithm to improve the efficiency of the algorithm. The algorithm randomly exchanges parts of two candidate solutions in the adaptive neighbourhood. The core idea is to generate a new solution through the characteristics of the optimal solution. Wu et al. (2019) solved a service selection problem by combining the genetic algorithm and the simulated annealing algorithm. The algorithm introduces the temperature parameters of simulated annealing into the genetic algorithm, which combines the advantages of the two algorithms.

The algorithm inherits the powerful global search ability and avoids falling into local optimisation. Zhang et al. (2010) proposed a composite agent service selection algorithm based on simulated annealing. Firstly, the algorithm finds a composite agent service with minimal cost, and takes it as the initial candidate solution. Then, the algorithm enters a simulated annealing phase and searches for better solutions. Alayed et al. (2019) proposed an enhancement of the ant colony algorithm. The crossover operator is applied in this algorithm to generate new offspring. Zhu et al. (2021) combined the crossover and variation factors in the genetic algorithm into the cuckoo optimisation algorithm to select the best service from numerous candidate services to complete the task and improve user satisfaction.

Mistry et al. (2018) used a hybrid genetic algorithm to solve infrastructure as a service (IaaS) problem, in this metaheuristic approach, a preferential heuristic is the probabilistic indicator of request's influence on the fitness. Dynamic solutions are generated by updating heuristic values at regular intervals. Jia et al. (2019) proposed three hybrid algorithms based on the SOA and heat exchange optimisation to solve the problem of feature selection (Jia et al., 2019). Firstly, the roulette algorithm is employed to update positions; secondly, the thermal exchange optimisation (TEO) algorithm is applied in the SOA algorithm; in the last step, the TEO algorithm's heat exchange formula is employed to improve the seagull attack mode of the SOA algorithm. Jiang et al. (2020) applied the opposition-based learning algorithm to the SOA to optimise the initial population of seagulls, so the accuracy and computational efficiency are improved.

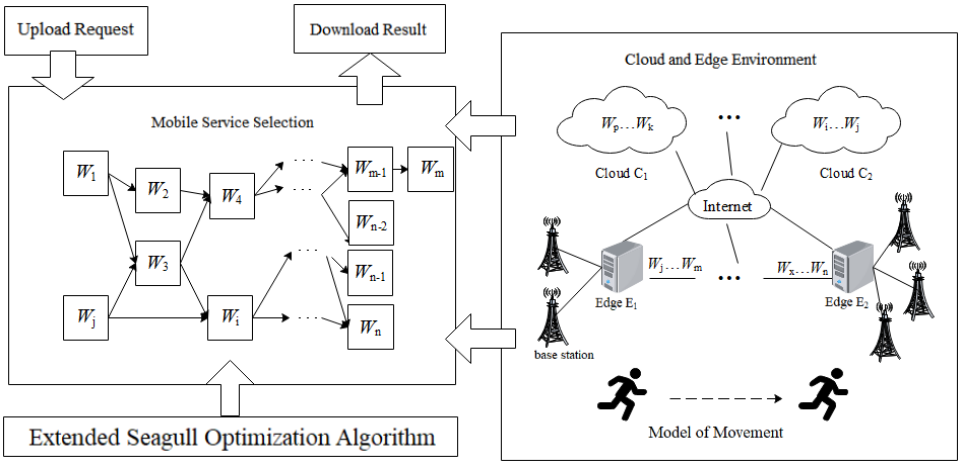
4 The proposed ESOA algorithm

This section provides more details of the proposed approach: the ESOA and fitness functions are designed to solve the mobile service selection problem.

Figure 5 illustrates components of the proposed approach as follows:

- The cloud and edge environment: this component mainly describes the distribution of services on cloud and edge servers. In addition, information for data transmission between base stations and users is captured. A user's moving path model is built.
- ESOA: this component describes the ESOA algorithm, which combines the SOA and the SA. Detailed information on the ESOA is described in Section 4.1.
- Mobile service selection: the ESOA algorithm is employed to solve the mobile service selection problem. Fitness functions are designed to calculate QoS values of seagull positions. The service combination with the best response time/cost is returned as the solution.

Figure 5 The proposed service selection approach



A mobile user uploads his request to a nearby base station; the request is uploaded to servers. Then, the ESOA algorithm is applied to select services. Finally, a solution is returned to the mobile user. In this paper, we make the following assumptions:

- A server is connected with one or more base stations and other servers. A base station is connected with an edge server and multiple cloud servers, and has overlapping coverage with other base stations.
- An edge server provides a small number of services, while a cloud server provides more services. The data transmission time from a base station to an edge server is less than that from the base station to a cloud server.
- A user submits a request to a base station that covers him/her, and a solution is sent back to the user.

Algorithm 1 Extended seagull optimisation algorithm

Input *population*: the population of seagull
maxGeneration: the maximum generations
Output P_{bs}: the optimal solution
procedure
Initialise A with equation (9)
Initialise BL with equation (11)
$f_c \leftarrow 2$
$u \leftarrow 1$
$k \leftarrow 1$
for iteration ← 1 to *maxGeneration* **do**
    Choose a position with the maximum fitness value as P_{bs}
    /*Calculate and choose the best seagull according to the fitness function*/
    for $i \leftarrow 1$ to population **do**
        /*Migration behaviour*/
        $rd \leftarrow Rand(0, 1)$
        $\alpha \leftarrow$ Rand$(0, 2\pi)$
        /*Attacking behaviour*/
        update h /*According to equation (16)*/
        update $L_s(t)$ /*According to equation (12)*/
        compute x, y, z /*According to equations (13)–(16)*/
        update P_{si} /*According to equation (17) */
        A new solution P_{ns} is generated by the SA algorithm
        choose the better one between P_{si} and P_{ns} as the new P_{bs}.
    end
end
return P_{bs} as the solution
end procedure

4.1 Extended seagull optimisation algorithm

The SOA is an evolutionary optimisation algorithm inspired by seagulls' migration and predation behaviours (Dhiman and Kumar, 2019). This algorithm has been widely used in various areas to solve optimisation problems (Dhiman et al., 2020). However, according to the research of Cao et al. (2019), the SOA has some disadvantages, such as reduced diversity, exploration ability and premature convergence. In order to overcome these disadvantages, we introduce the SA into the SOA, and name it the ESOA. In Algorithm 1, the ESOA algorithm is explained. The performance of the ESOA is tested in Section 6.

Algorithm 1 is the ESOA algorithm, where the SA (Bangert, 2012) is employed. Position P_s represents a candidate solution. Specifically, in each iteration, a seagull i generates a position P_{si} by the SOA (line 21), and another position P_{ns} is generated by the SA (line 22). Then, the better one between P_{ns} and P_{si} is stored as the optimal solution P_{bs} (line 23).

4.2 The fitness function

The mobile service selection problem in terms of response time/cost is modelled as an optimisation problem. In the proposed ESOA, fitness functions are designed to determine whether a seagull should contribute to the next generation. The fitness value depends on the response time/cost of the candidate solution. The smaller the value, the better the solution.

Supposing there are n tasks, s_i is selected for task i. Based on equations (1)–(3), the total cost of a solution is the sum of selected services' cost.

$$C_{\text{total}} = \sum_{i=1}^{n} C(s_i) \tag{19}$$

The overall response time T_{overall} is the total time spent from submitting a request to receiving a solution.

$$T_{\text{overall}} = T_{\text{up}} + R_{\text{comp}}{}_{i=1}^{n}(s_i) + T_{\text{down}} \tag{20}$$

where T_{up} is the time for uploading a request, R_{comp} is the response time of services combination, and T_{down} is time for downloading a solution.

R_{comp} is calculated based on the combination structure [equations (4)–(6)] of selected services for solving a request.

For T_{up}, we consider the following scenarios: if service s_1 for the first task is located on edge server e_i, the nearby base station receiving a user's request is connected with edge server e_j. The uploading time T_{up} includes the time spent from the user to the base station T_{p2b}, the data transferring time from the base station to the edge server T_{b2s}, and the time of transferring data $R(e_j, e_i)$ from e_j to e_i. If s_1 is located on a cloud server, the uploading time T_{up} is the total time for uploading to the base station T_{p2b}, and the time of transferring data from the base station to the cloud server T_{b2s}. The uploading time T_{up} is expressed as:

$$T_{\text{up}} = \begin{cases} T_{\text{p2b}} + T_{\text{b2s}} + R(e_j, e_i), & \text{if } s_1 \in e_i \\ T_{\text{p2b}} + T_{\text{b2s}}, & \text{if } s_1 \in c_k \end{cases} \tag{21}$$

Similarly, for the downloading time T_{down}, we consider the following scenarios: If service s_n for the last task is located on edge server e_j, the nearby base station that sends data to the user is connected with edge server e_i. The downloading time T_{down} includes the time of transferring data $R(e_j, e_i)$ from e_j to e_i, the time spent from the edge server to the base station T_{b2s}, and the time spend from the base station to the user T_{b2p}. If s_n is located on a cloud server, the downloading time T_{down} is the total time of transferring data from the cloud server to the base station T_{b2s}, and the downloading time T_{b2p} from the base station to the user. The downloading time T_{down} is expressed as:

$$T_{\text{down}} = \begin{cases} R(e_j, e_i) + T_{\text{b2s}} + T_{\text{b2p}}, & \text{if } s_n \in e_i \\ T_{\text{b2s}} + T_{\text{b2p}}, & \text{if } s_n \in c_k \end{cases} \tag{22}$$

The time of uploading a request to a base station T_{p2b} and the time of downloading a result from a base station T_{b2p} are calculated as follows:

$$T_{\text{p2b}} = \sum Task / r(B, g), \tag{23}$$

$$T_{\text{b2p}} = Result / r(B, g), \tag{24}$$

where $\sum Task$ means the data size of all tasks in the request, *Result* means the data size of downloading, and $r(B, g)$ represents achievable data-transfer rate. According to the Shannon-Hartley formula (Lazarakis et al., 1994), the bandwidth B and signal-to-noise ratio (SNR) determine the achievable data-transfer rate $r(B, g)$, and $r(B, g)$ can be calculated by equation (25).

$$r(B, g) = B * \log_2 (1 + tp * g / \sigma^2) \tag{25}$$

where σ^2 is the noise power at the receiver, tp is wireless transmit power of the mobile device, g represents the channel power gain, and can be calculated as $h_0 * d^{-\gamma}$, where h_0 is the received power, d is the distance between the mobile device and the base station, and $\gamma = 2$ is the path loss factor (Chen et al., 2016).

The transmission time T_{b2s} between a base station and a server depends on the size of data to be transferred and the bandwidth B_{b2s}, which is calculated as follows:

$$T_{\text{b2s}} = DataSize / B_{\text{b2s}} \tag{26}$$

The transmission time R between two servers e_i and e_j depends on the size of data to be transferred and the bandwidth B_{e2e}, which is calculated as follows:

$$R(e_i, e_j) = DataSize / B_{e2e} \tag{27}$$

5 Case study

In this section, we give a simple but meaningful example to show how the ESOA is applied to solve a mobile service selection problem and illustrate the necessity and significance of our work.

In this scenario, Sam's job is to deliver food from a restaurant to a customer. When a delivery order is received, Sam needs to do facial recognition and upload a video of his

current environment to confirm that he is ready for the order. After that, Sam receives a route plan from his position to the restaurant and from the restaurant to the customer. Besides, the route plan provides photos of the restaurant.

As shown in Figure 6, once a delivery order arrives, three tasks are generated and executed sequentially, including facial recognition task ($Task_1$), uploading environment video task ($Task_2$), and route plan task ($Task_3$). Supposing each task has four candidate services, information of candidate services is available (Table 3). Sam's delivery route and the network condition along the way are predetermined. The whole delivery path is within the wireless transmission range of the same cloud server. The cloud server and edge servers are connected with each other. Candidate services are randomly distributed on cloud and edge servers.

Figure 6 The delivery route of Sam (see online version for colours)

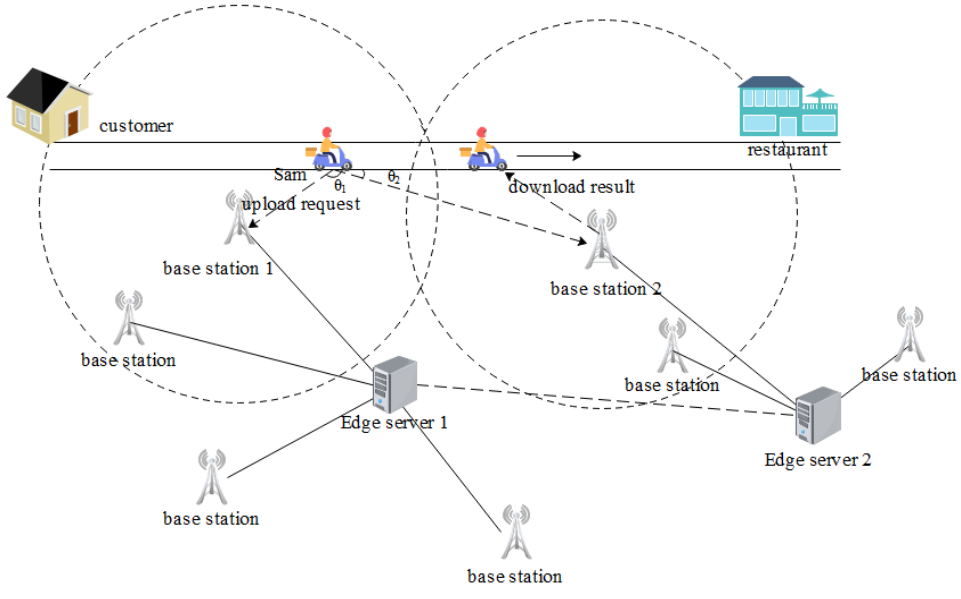


Table 2 Tasks and candidate services

Task	*Candidate service*			
Face recognition task ($Task_1$)	rec_1	rec_2	rec_3	rec_4
Uploading environment video task ($Task_2$)	$video_1$	$video_2$	$video_5$	$video_6$
Route plan task ($Task_3$)	$route_1$	$route_3$	$route_7$	$route_9$

Table 3 QoS values of candidate services

Service	*Response*	*Cost*	*Service*	*Response*	*Cost*	*Service*	*Response*	*Cost*
rec_1	156 ms	24	$video_1$	382 ms	168	$route_1$	548 ms	268
rec_2	180 ms	60	$video_2$	351 ms	188	$route_3$	506 ms	298
rec_3	102 ms	51	$video_5$	310 ms	122	$route_7$	518 ms	208
rec_4	153 ms	47	$video_6$	402 ms	176	$route_9$	580 ms	231

Table 2 shows tasks and their candidate services. Table 3 shows QoS values of services. Table 4 shows distribution of services on servers. For example, the first row means that services rec_1, $video_5$ and $route_3$ are deployed on edge server e_1.

Sam needs to perform facial recognition and upload a video of his environment, $Task_1$ represents facial recognition, $Task_2$ contains a video of Sam's environment, and $Task_3$ represents route plan. When the tasks are completed, Sam receives a route to the restaurant, from the restaurant to the customer, and photos of the restaurant and the food, the size of` result is about 10 Mb. The speed of Sam is 15 m/s. Supposing when Sam receives the order, the distance between Sam and base station 1 (resp. base station 2) is 100 m (resp. 200 m), and the angle between Sam and base station 1 (resp. base station 2) is 135° (resp. 30°). We assume that the angle between Sam's moving direction and the base station remains unchanged; the signal range of a base station is 150 m. Values of parameters are listed in Table 5.

Table 4 Services deployment on servers

Server	*Services*				
e_1	rec_1	$video_5$	$route_3$		
e_2	rec_3	rec_4	$video_1$	$route_7$	
c	rec_2	$video_2$	$video_6$	$route_1$	$route_9$

Table 5 Parameters of the example

Parameter	*Value*	*Parameter*	*Value*
$Task_1$	9 Mb	$Task_3$	1 Mb
$Task_2$	40 Mb	*Result*	10 Mb
B	256 MHz	tp	500 MW
h_0	–30 dB	σ^2	–60 dBm/Hz
B_{e2c}	2.5 Gbps	B_{b2e}	2 Gbps
B_{e2e}	3 Gbps	B_{b2c}	1.5 Gbps
v_s	15 m/s	λ	2
d_1	100 m	d_2	200 m
θ_1	135°	θ_2	30°

5.1 Time of uploading

The request of Sam is received by the nearest base station. According to equation (23) and equation (25), time of uploading the request to base station 1 is:

$$\begin{aligned} T_{p2b} &= \sum Task / r(B, g) \\ &= \sum Task / B * \log_2 \left(1 + tp * g / \sigma^2\right) \\ &= 5,486\ ms \end{aligned}$$

According to equation (26), the time of uploading the request to e_1 is:

$$T_{b2e_1} = DataSize / B_{b2e}$$
$$= 25\ ms$$

The initial seagull position is randomly selected as $P_s(t) = \{rec_1, video_1, route_1\}$, the optimal position of the initialisation population is P_{bs} is $\{rec_1, video_2, route_3\}$, according to equation (4) and equation (27), the response time of $P_s(t)$ is:

$$R_{comp} = R(rec_1) + R(e_1, e_2) + R(video_1) + R(e_2, c) + R(route_1)$$
$$= 156 + 41/3 + 382 + 1/2.5 + 548 \approx\sim 1{,}100.1\ ms$$

Supposing the maximum number of iterations $Max_{iteration} = 100$, $f_c = 2$, according to equation (9).

$$A = f_c - \left(t \times \left(f_c / Max_{iteration}\right)\right) = 1.98$$

According to equation (8), we get the new position $P_{ns}(t)$:

$$P_{ns}(t) = A \times P_s(t) = \{rec_2, video_2, route_2\}$$

Assuming $rd = 0.1275$, according to equation (11), BL is calculated as follows:

$$BL = 2 \times A^2 \times rd = 1$$

According to equation (10) and equation (12), we get $M_s(t)$ and $L_s(t)$:

$$M_s(t) = BL \times \left(P_{bs}(t) - P_s(t)\right) = \{rec_0, video_1, route_2\}$$

$$L_s(t) = \left|P_{ns}(t) + M_s(t)\right| = \{rec_2, video_3, route_4\}$$

Supposing randomly select $\alpha = \frac{\pi}{6}$, $u = 1$, $k = 1$, according to equation (13) to equation (16), x, y, z, h are calculated as follows:

$$h = u \times e^{\alpha k} = e^{\frac{\pi}{6}}$$

$$x = h \times \cos(\alpha) = \frac{\sqrt{3}}{2} e^{\frac{\pi}{6}}$$

$$y = h \times \sin(\alpha) = \frac{1}{2} e^{\frac{\pi}{6}}$$

$$z = h \times \alpha = \frac{\pi}{6} e^{\frac{\pi}{6}}$$

Finally, according to equation (17), the updated position of SOA is obtained:

$$P_s(t) = L_s(t) \times x \times y \times z + P_{bs}(t) = \{rec_3, video_5, route_7\}$$

As rec_3 is located on e_2, according to equation (22), the upload time is:

$$T_{up} = T_{p2b} + T_{b2s} + R(e_1, e_2) = 5.527.66\ ms$$

The simulated annealing algorithm is applied in the ESOA, assuming initially T_{sa} = 10 and the temperature drop rate dT = 0.01, $T_{\min}$ = 0.5, $r_{\max}$ = 1. Supposing random numbers which modify services for three tasks are 0.56, 0.42, 0.62, according to equation (18).

$$S_s(t) = P_s(t) + (2 \times f - r_{\max}) \times T_{sa} = \{rec_4, video_5, route_9\}$$

After several iterations, when the temperature cools, T_{sa} = 1, the solution $S_s(t)$ = {rec_4, $video_5$, $route_9$} is returned.

5.2 Time of service combination

According to equation (4) and equation (27), time spent on service combination of $P_s(t)$ is:

$$\begin{aligned} R(P_s(t)) &= R(rec_3) + \sum_{Task_2}^{Task_3} R(e_2, e_1) + R(video_5) + R_{task_3}(e_1, e_2) + R(route_7) \\ &= 102 + 41/3 + 310 + 0.33 + 518 \approx\sim 944\ ms \end{aligned}$$

Time spent on service combination of $S_s(t)$ is:

$$R(S_s(t)) = R(rec_4) + \sum_{Task_2}^{Task_3} (e_2, e_1) + R(video_5) + R_{task_3}(e_1, c) + R(route_9) = 1{,}057\ ms$$

5.3 Time of downloading

When the solution is ready, according to equation (7), the distance between Sam and base station 1 (resp. base station 2) is 182.46 m (resp. 125.43 m), therefore, base station 2 is chosen as it is nearer to Sam, the solution is sent from base station 2 to Sam. For $P_s(t)$, the last service is located on edge server 2, the result is sent to base station 2.

$$\begin{aligned} T_{b2s} &= DataSize / B_{b2e} \\ &= 5\ ms \end{aligned}$$

Time for downloading from base station 2 is

$$\begin{aligned} T_{b2p} &= Result / r(B, g) \\ &= Result / B * \log_2(1 + tp * g / \sigma^2) \\ &= 1{,}721\ ms \end{aligned}$$

For $S_s(t)$, the last service is located on the cloud server, and the result is sent to base station 2.

$$\begin{aligned} T_{b2s} &= DataSize / B_{b2c} \\ &\approx 6.67\ ms \end{aligned}$$

The time for downloading from base station 2 is

$$T_{b2p} = Result / r(B, g)$$
$$= Result / B * \log_2 (1 + tp * g / \sigma^2)$$
$$= 1,721\ ms$$

The overall response time of $P_s(t)$ is

$$T = T_{p2b} + T_{b2s} + R(e_1, e_2) + R_{comp} + T_{b2s} + T_{b2p}$$
$$= 8,197.67\ ms$$

The overall response time of $S_s(t)$ is

$$T = T_{p2b} + T_{b2s} + R(e_1, e_2) + R_{comp} + T_{b2s} + T_{b2p}$$
$$= 8,312.33\ ms$$

In this example, the optimal position is $\{rec_3, video_5, route_7\}$, and the overall response time is 8,197.67 ms.

6 Experiment

This section describes experimental setup and performs experiments to investigate the performance of the ESOA against other algorithms.

6.1 Experiment setup

Algorithms are implemented in Java and run on a 1.9 GHz Intel Core i7 processor with Windows 10 64-bit operating system. We compare the ESOA with other evolutionary algorithms; the referencing algorithms are presented in Table 6.

Table 6 Algorithms used in experiments

Algorithm	*Description*
Seagull optimisation algorithm (SOA) (Dhiman and Kumar, 2019)	Imitates seagulls' migration and hunting behaviours
Genetic algorithm (GA) (Whitley, 1994)	Genetic principles: crossover and mutation
Simulated annealing (SA) (Bangert, 2012)	Temperature parameters of metal annealing
Ant colony optimisation (ACO) (Dorigo et al., 2006)	Imitates foraging behaviours of ants
Differential evolution(DE) (Price and Price, 1997)	Mutation generated by differences of parents
Particle swarm optimisation (PSO) (Price and Storn, 1997)	Behaviours of bird flocking or fish schooling

We obtain geographic information of base stations from the Australian Communication and Media Authority dataset (Lai et al., 2018). We assume that several base stations are equipped with an edge server. Services are distributed randomly on cloud and edge servers. We set the loss parameters in the process of information transmission: the response time between two edge servers varies from 1 ms to 10 ms, the response time between an edge server and cloud server varies from 10 ms to 30 ms, and the response

time between two cloud servers varies from 30 ms to 50 ms. Default values of parameters involved in the experiment are shown in Table 7.

Table 7 Parameter setting

Parameter	*Value*	*Parameter*	*Value*
Uploading request	50 Mb	B	256 MHz
Downloading result	10 Mb	v_{min}	1.1 m/s
d_{min}	10 m	v_{max}	1.5 m/s
d_{max}	200 m	h_0	–30 dB
tp	500 mW	σ^2	–60 dBm/Hz
B_{b2s}	1 Gbps		

Four datasets are generated with different number of tasks, each task has 100 candidate services (Table 8).

Table 8 Datasets

	Dataset1	*Dataset2*	*Dataset3*	*Dataset4*
Tasks	100	110	120	130
Candidate services for each task	100	100	100	100

Table 9 Experiment Results with the minimal overall response time (ms)

		Dataset1	*Dataset2*	*Dataset3*	*Dataset4*
ESOA	Response time	*3,005.3*	*3,415.4*	*3,860.9*	*4,291.9*
	At iteration	8	8	7	9
SOA (Dhiman and Kumar, 2019)	Response time	3,025.9	3,435.0	3,881.3	4,319.6
	At iteration	6	6	6	6
GA (Whitley, 1994)	Response time	15,176.5	16,723.6	22,546.8	24,718.1
	At iteration	96	94	99	99
SA (Bangert, 2012)	Response time	6,658.5	7,838.6	8,654.0	10,632.5
	At iteration	68	33	13	11
ACO (Dorigo et al., 2006)	Response time	3,581.8	3,862.1	4,319.6	4,535.2
	At iteration	4	7	4	10
DE (Price and Storn, 1997)	Response time	6,064.2	6,957.1	8,153.3	8,809.0
	At iteration	83	89	96	93
PSO (Eberhart and Shi, 2001)	Response time	5,107.4	6,733.0	7,151.4	8,875.6
	At iteration	52	59	81	64

6.2 Analysis of results

We compare the performance of the ESOA against other evolutionary algorithms referenced in this paper. In the experiment, we set the maximum generations to 100; services are randomly distributed on edge/cloud servers. Table 9 shows the performance of the algorithms in terms of the overall response time. 'At iteration' means the best result is generated at which iteration. The overall response time is a negative criterion, the

smaller the value, the better the solution. As we can see from this table, the performance of the ESOA is better than other algorithms. The convergence rate of the SOA, the ACO and the ESOA is much faster than the GA, the SA, the DE and the PSO algorithms.

Figure 7 Response time and iterations (see online version for colours)

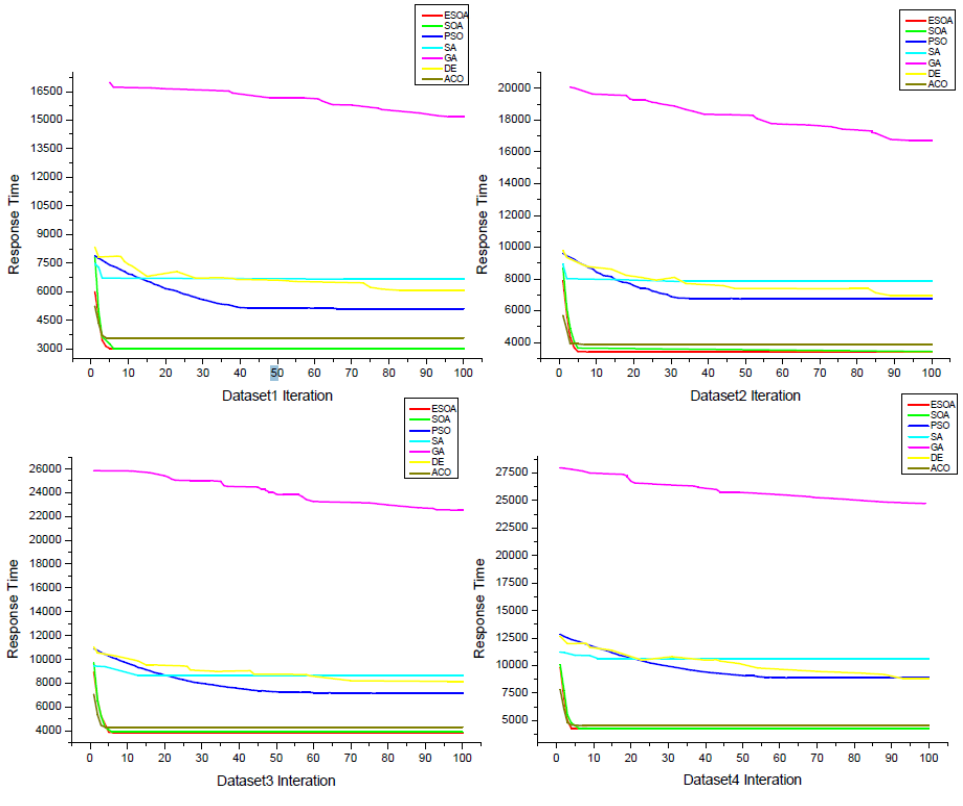


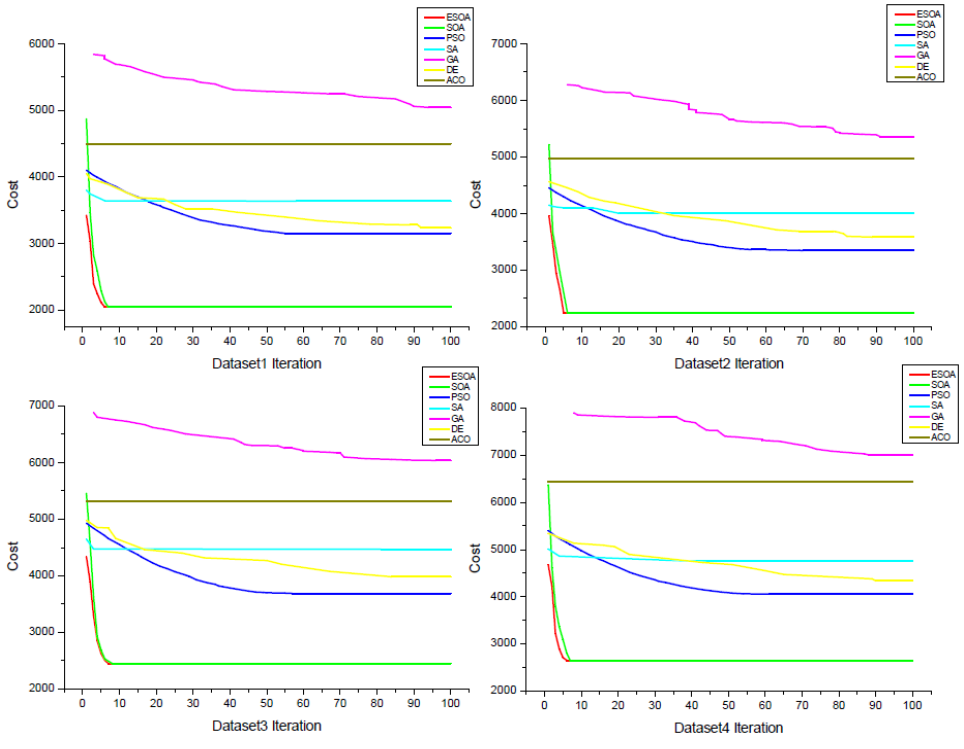
Table 10 Experiment results with the minimal cost (cents)

		Dataset1	*Dataset2*	*Dataset3*	*Dataset4*
ESOA	Cost	*2,047*	*2,249*	*2,451*	*2,646*
	At iteration	6	6	6	7
SOA (Dhiman and Kumar, 2019)	Cost	*2,047*	*2,249*	*2,451*	*2,646*
	At iteration	7	7	8	9
GA (Whitley, 1994)	Cost	5,046	5,363	6,041	7,006
	At iteration	94	91	96	88
SA (Bangert, 2012)	Cost	3,638	4,003	4,469	4,752
	At iteration	58	20	81	40
ACO (Dorigo et al., 2006)	Cost	4,500	4,971	5,314	6,438
	At iteration	1	1	1	1

Table 10 Experiment results with the minimal cost (cents) (continued)

		Dataset1	*Dataset2*	*Dataset3*	*Dataset4*
DE (Price and Storn, 1997)	Cost	3,237	3,583	3,983	4,337
	At iteration	92	89	82	90
PSO (Eberhart and Shi, 2001)	Cost	3,153	3,344	3,687	4,050
	At iteration	56	71	60	61

Figure 8 Cost and iterations (see online version for colours)



We also investigate the performance in terms of the cost. From Table 10, the ESOA and the SOA can find a solution with less cost. Besides, the ESOA can find a solution with the same price as the SOA, but in fewer iterations. Although the ACO algorithm can find a solution with only one generation, the cost is more than twice than that of the ESOA and the SOA algorithms.

Figure 7 shows the relationship between the response time and iterations. It can be seen that the ESOA can find a solution with less response time.

Figure 8 shows the relationship between iterations and cost. As can be seen from this figure, the ESOA and the SOA can find solutions with less cost than other algorithms in fewer iterations. Time spent on searching of the EOSA algorithm is less than that of the SOA.

Figure 9 Response time and iterations (see online version for colours)

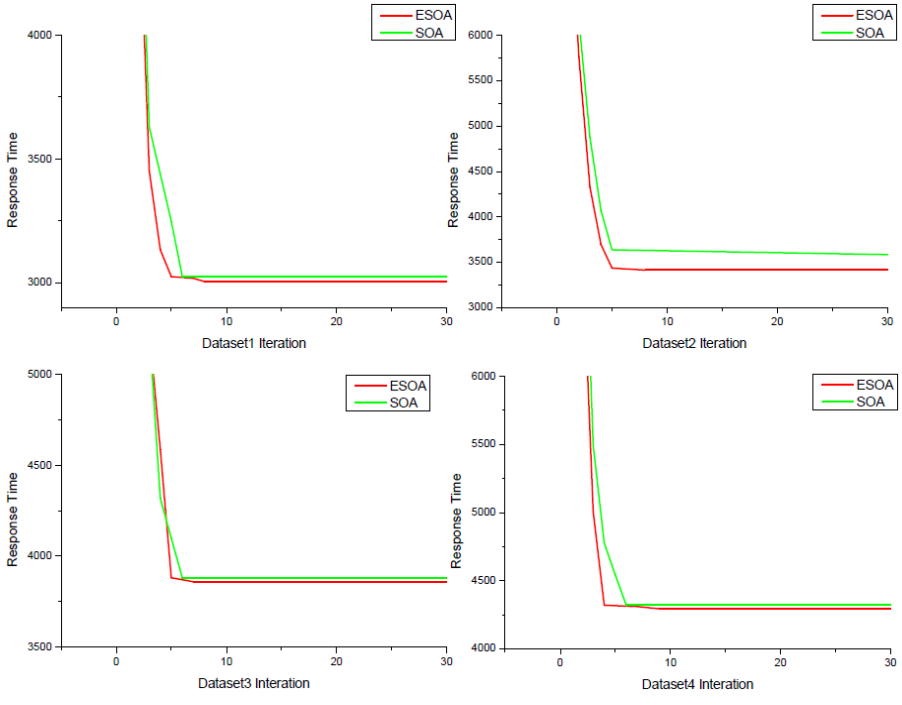
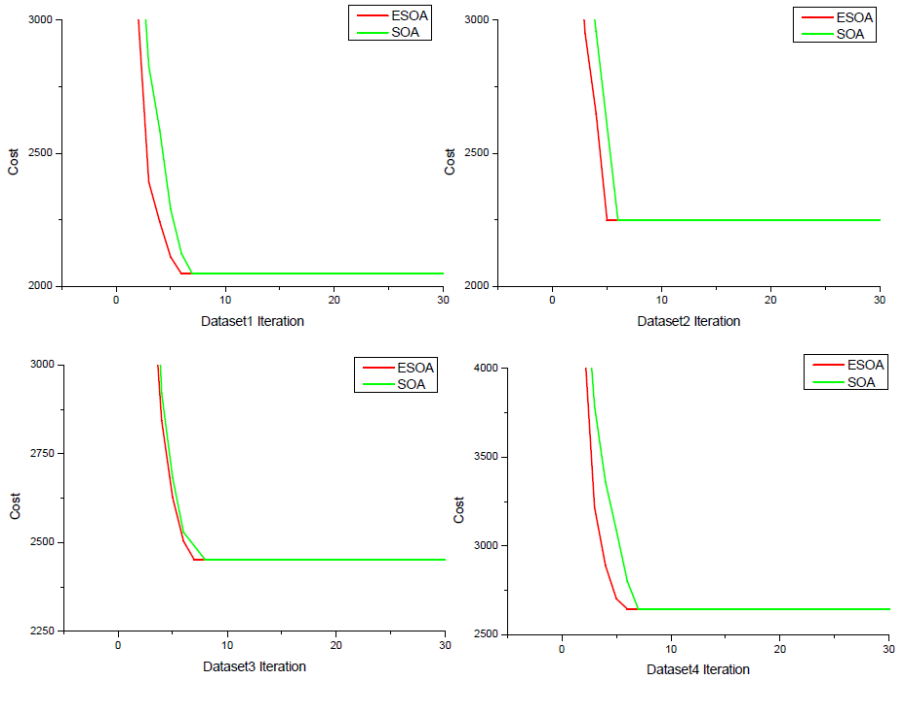


Figure 10 Cost and iterations (see online version for colours)



As the ESOA and the SOA converge much faster than other selected methods, we enlarge the scales of response time and cost so that differences between the ESOA and the SOA can be see clearer. When the goal is to find a solution with cheapest price, although the ESOA and the SOA return solutions with same price, the ESOA converges faster than the SOA algorithm. Figure 9 and Figure 10 show more detailed information of the ESOA and the SOA in terms of response time and cost respectively.

The above-mentioned experimental results show that ESOA can efficiently and effectively find a solution, and outperforms other five algorithms.

7 Conclusions and future work

We propose to combine seagull optimisation and simulated annealing algorithm to solve the mobile service selection problem in the cloud and edge computing environment. Simulated annealing algorithm is applied to avoid falling into local maximum and premature convergence problem. Additionally, the random waypoint mobility model is applied to simulate the mobile path of the user. Finally, experimental results show that the proposed ESOA algorithm outperforms six other evolutionary algorithms in terms of QoS values and convergence rate. In our next work, we will put our proposed approach into real life and do experiments with real services. Taking energy consumption into consideration would be an interesting extension. We may consider energy consumption of mobile devices and base stations. The energy consumption of a base station includes data transmission and data processing energy (Xu et al., 2022). The former one is related to the amount of data to be transferred, and the latter one is related to the rate of accessing its processing unit and peak power. The energy consumed by processing the request in a base station consists of the energy consumption of its processing unit, idle power, and leakage power.

References

Bangert, P. (2012) 'Optimization: simulated annealing', *Optimization for Industrial Problems*, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-24974-7_7.

Bettstetter, C., Hartenstein, H. and Perez-Costa, X. (2004) 'Stochastic properties of the random waypoint mobility model', *Wireless Networks*, Vol. 10, No. 5, pp.555–567.

Cao, Y., Li, Y., Zhang, G., Jermsittiparsert, K. and Razmjooy, N. (2019) 'Experimental modeling of PEM fuel cells using a new improved seagull optimization algorithm', *Energy Reports*, Vol. 5, pp.1616–1625, https://doi.org/10.1016/j.egyr.2019.11.013.

Chen, X., Jiao, L., Li, W. and Fu, X. (2016) 'Efficient multi-user computation offloading for mobile-edge cloud computing', *IEEE/ACM Transactions on Networking: A Joint Publication of the IEEE Communications Society, the IEEE Computer Society, and the ACM with Its Special Interest Group on Data Communication*, Vol. 24, No. 5, pp.2795–2808.

Dahan, D., Hassan, M. and Mohammed, A. (2019) 'Two-step artificial bee colony algorithm enhancement for QoS-aware web service selection problem', *IEEE Access*, Vol. 7, pp.21787–21794 [online] https://ieeexplore.ieee.org/document/8625404.

Deng, S., Zhao, H., Fang, W., Yin, J., Schahram, D. and Albert, Y.Z. (2020) 'Edge intelligence: the confluence of edge computing and artificial intelligence', *IEEE Internet of Things Journal*, Vol. 7, No. 8, pp.7457–7469.

Dhiman, G. and Kumar, V. (2019) 'Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems', *Knowledge-Based Systems*, February 1, Vol. 165, pp.169–196, https://doi.org/10.1016/j.knosys.2018.11.024.

Dhiman, G., Singh, K.K., Slowik, A., Chang, V. and Garg, M. (2020) 'Emosoa: a new evolutionary multi-objective seagull optimization algorithm for global optimization', *International Journal of Machine Learning and Cybernetics*, Vol. 12, pp.571–596, https://doi.org/10.1007/s13042-020-01189-1.

Dorigo, M., Birattari, M. and Stutzle, T. (2006) 'Ant colony optimization', *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp.28–39.

Eberhart, R. and Shi, Y. (2001) 'Particle swarm optimization: developments, applications and resources', in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, Vol. 1, pp.81–86.

Gelenbe, E. and Lent, R. (2013) 'Energy-qos trade-offs in mobile service selection', *Future Internet*, Vol. 5, No. 2, pp.128–139.

Hashem, A., Fadl, D., Taha, A., Hassan, M. and Mohammed, A. (2019) Enhancement of ant colony optimization for qos-aware web service selection', *IEEE Access*, Vol. 7, pp.97041–97051 [online] https://ieeexplore.ieee.org/document/8758409.

Hong, C.H. and Varghese, B. (2019) 'Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms', September, *ACM Comput. Surv.*, Vol. 52, No. 5, Article No. 97, pp.1–37, https://doi.org/10.1145/3326066.

Huang, J., Lan, Y. and Xu, M. (2018) 'A simulation-based approach of QoS-aware service selection in mobile edge computing', *Wireless Communications and Mobile Computing*, Vol. 2018, Article ID 5485461, 10pp, https://doi.org/10.1155/2018/5485461.

Jia, H., Xing, Z. and Song, W. (2019) 'A new hybrid seagull optimization algorithm for feature selection', *IEEE Access*, Vol. 7, pp.2169–3536 [online] https://ieeexplore.ieee.org/document/8684847.

Jiang, H., Yang, Y., Ping, W. and Dong, Y. (2020) 'A novel hybrid classification method based on the opposition-based seagull optimization algorithm', *IEEE Access*, Vol. 8, pp.100778–100790 [online] https://ieeexplore.ieee.org/document/9099867.

Lai, P., He, Q., Abdelrazek, M., Chen, F., John, H., John, G. and Yang, Y. (2018) 'Optimal edge user allocation in edge computing with variable sized vector bin packing', in *International Conference on Service-Oriented Computing*, pp.230–245.

Lazarakis, F., Tombras, G.S. and Dangakis, K. (1994) 'Average channel capacity in a mobile radio environment with Rician statistics', *IEICE Transactions on Communications*, Vol. E77-B, No. 7, pp.971–977.

Li, Y. and Wang, S. (2018) 'An energy-aware edge server placement algorithm in mobile edge computing', in *2018 IEEE International Conference on Edge Computing (EDGE)*, pp.66–73.

Ma, X., Wang, S., Zhang, S., Yang, P., Lin, C. and Shen, X. (2021) 'Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing', *IEEE Transactions on Cloud Computing*, Vol. 9, No. 3, pp.968–980.

Ma, X., Zhang, S., Li, W., Zhang, P., Lin, C. and Shen, X. (2017) 'Cost-efficient workload scheduling in cloud assisted mobile edge computing', in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp.1–10.

Mach, P. and Becvar, Z. (2017) 'Mobile edge computing: a survey on architecture and computation offloading', *IEEE Communications Surveys Tutorials*, Vol. 19, No. 3, pp.1628–1656.

Mistry, S., Bouguettaya, A., Dong, H. and Qin, A.K. (2018) 'Metaheuristic optimization for long-term IaaS service composition', *IEEE Transactions on Services Computing*, January, Vol. 11, No. 1, pp.131–143.

Muhammad, A., Wang, Y., Wang, K. and Pei-Qiu, H. (2020) 'A review on computational intelligence techniques in cloud and edge computing', *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 4, No. 6, pp.742–763.

Orsini, G., Bade, D. and Lamersdorf, W. (2016) 'Cloudaware: a context-adaptive middleware for mobile edge and cloud computing applications', in *IEEE International Workshops on Foundations & Applications of Self Systems*.

Price, K.V. and Storn, R. (1997) 'Differential evolution-a simple evolution strategy for fast optimization', *Journal of Global Optimization*, Vol. 11, pp.341–359, https://doi.org/10.1023/A:1008202821328.

Raoul, H., Marco, G. and Sebastian, L. (2020) 'Making cloud service selection easy for SMEs: a tool for selecting saas services', in Hofmann, S., Müller, O. and Rossi, M. (Eds.): *Designing for Digital Transformation Co-Creating Services with Citizens and Industry*, pp.333–338, Springer International Publishing, Cham.

Reiff-Marganiec, S. and Tilly, M. (2008) 'Non-functional property based service selection: a survey and classification of approaches', in *Proceedings of the 2008 Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop*.

Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016) 'Edge computing: vision and challenges', *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp.637–646.

Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A. and Shen, X. (2021) 'Delay aware microservice coordination in mobile edge computing: a reinforcement learning approach', *IEEE Transactions on Mobile Computing*, Vol. 20, No. 3, pp.939–951.

Wang, T., Zhang, G., Liu, A., Bhuiyan, M.Z.A. and Jin, Q. (2019) 'A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing, *IEEE Internet of Things Journal*, Vol. 6, No. 3, pp.4831–4843.

Whitley, D. (1994) 'A genetic algorithm tutorial', *Statistics & Computing*, Vol. 4, No. 2, pp.65–85.

Wu, H., Deng, S., Li, W., Yin, J. and Zomaya, A.Y. (2019) 'Mobility-aware service selection in mobile edge computing systems', in *2019 IEEE International Conference on Web Services (ICWS)*.

Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S. and Qi, L. (2019) 'A computation offloading method over big data for IoT-enabled cloud-edge computing', *Future Generation Computer Systems*, Vol. 95, pp.522–533, https://doi.org/10.1016/j.future.2018.12.055.

Xu, Z., Zhou, L., Dai, H., Liang, W., Zhou, W., Zhou, P., Xu, W. and Wu, G. (2022) 'Energy-aware collaborative service caching in a 5g-enabled MEC with uncertain payoffs', *IEEE Transactions on Communications*, Vol. 70, No. 2, pp.1058–1071.

Zhang, J., Xia, W., Cheng, Z., Zou, Q., Huang, B., Shen, F., Yan, F. and Shen, L. (2017) 'An evolutionary game for joint wireless and cloud resource allocation in mobile edge computing', in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp.1–6.

Zhang, K., Zhang, H., Jiang, L. and Xu, J. (2010) 'Composite agent service selection algorithm for non-functional attributes based on simulated annealing', in *2010 Second World Congress on Software Engineering*, Vol. 2, pp.101–106.

Zhu, M., Yu, F., Yan, X., Li, J. and Wang, Y. (2021) 'Scaling up mobile service selection in edge computing environment with cuckoo optimization algorithm', in *2021 IEEE International Conference on Services Computing (SCC)*, pp.394–400.