Energy-aware edge server placement using the improved butterfly optimization algorithm

Ali Asghari (Asghari_ali@aut.ac.ir)
Shafagh Institute of Higher Education
Marjan Sayadi
Shafagh Institute of Higher Education
Hossein Azgomi
Islamic Azad University

Research Article

Keywords: Server placement, BOA, CRO, Energy, Latency

Posted Date: September 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2071513/v1

Energy-aware edge server placement using the improved butterfly optimization algorithm

Ali Asghari (Corresponding Author)
Department of computer engineering, Shafagh Institute of Higher Education, Tonekabon, Iran
Asghari_ali@aut.ac.ir

Marjan Sayadi
Department of computer engineering, Shafagh Institute of Higher Education, Tonekabon, Iran
msayadi@shafagh.ac.ir

Hossein Azgomi
Department of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Iran
Azgomi@iaurasht.ac.ir

Abstract

Cloud service providers transfer some of their resources to the proximity of their users in order to increase the quality of services provided to them. Proper placement of servers, considering the number of service demands in different parts of the network, not only plays an important role in providing better services to users but also causes more effective use of resources and reduces their energy consumption. Some related research has been done in this context. However, designing a model that can meet the needs of both the users and the service providers has received less attention. On the other hand, most researchers use discrete models to select a number of candidate locations for resource deployment, while the proposed method explores the entire search area to find optimal locations for server placement. The proposed method (ESPB) using butterfly optimization algorithm(BOA), DVFS technic, and coral reefs optimization algorithm(CRO) seeks to find the best locations for edge servers. In the first step, BOA is used to find the best locations for resource deployment. Then the CRO algorithm is used to map between the optimal locations and the servers. The experiments show that the proposed method can effectively save energy and reduces network latency.

Keywords: Server placement, BOA, CRO, Energy, Latency

1. Introduction:

Cloud computing is considered a popular model for providing various services offered to users. In this model, cloud users can access the services anywhere and anytime [1]. The emergence of the 5G cellular communication technology and the widespread use of smart communication equipment led to the formation of new needs in this field. New applications include the Internet of Things, online games, social networks, and electronic banking, which must consider users' mobility [2]. Due to the hardware limitation of mobile smart equipment (processing power, battery, and memory), most of the services required by users should be offloaded on network servers [3]. Therefore, the traditional models of cloud computing quickly adapted to these conditions and with the aim of providing high-quality services, transferred some of their resources to the edge of the network and in the vicinity of users [4]. However, due to the number of processes and volume of data exchanged, more effective use of resources plays an important role in providing suitable services to users.

Proper server placement has an important role in improving service quality, and consequently, users will access the most suitable resources they need. Otherwise, server overloading and underloading may occur

in different parts of the network [5]. Cloud service providers, while increasing the efficiency of their services, prefer to reduce the energy consumption of servers, which consequently reduces costs [6]. On the other hand, users need services with tolerable latency. For this reason, the proposed method considers the interests of both service providers and users. In most research, some predefined candidate locations have been provided for server placement [7]. This approach is due to the assumption of the discrete nature of the problem. The proposed method ignores this limitation and explores the entire search space to find the optimal resource placement locations. In the proposed method and in the first step, the butterfly optimization algorithm [8] is used to find the optimal location of the servers. Each butterfly is considered a solution. The attractiveness of a solution is proportional to its value according to the objective function of the problem. In the second phase of the proposed method, the coral reefs optimization algorithm [9] is used to find the best mapping between servers and the optimal locations obtained in the previous stage. optimal server placement can lead to both optimizing the server's energy consumption and the resource access delay reduction. The DVFS technique [10] has been used to reduce resource energy consumption. Based on this technique, a server has different power states, and a task can be executed with its minimum possible power mode.

The contributions of the proposed method are as follows:

- Using the butterfly optimization algorithm to find the best location of the servers in each area
- Utilization of the coral reefs optimization algorithm to allocate the best servers in each deployment location
- Using the DVFS technique to save energy consumption of resources
- Exploring the entire search space to find the optimal resource placement locations
- Considering the interests of both service providers and users

Other sections of the paper are as follows:

In section two, a review of related research is conducted. In section three, the basics used in the proposed method are described. In section four, the proposed method is introduced. Section five is related to the evaluation of the proposed method. Finally, the paper concludes in section six.

2. literature review

Edge server placement as a new paradigm in mobile cloud computing has recently been considered by some researchers. Due to the fact that the problem of server placement is an optimization problem, some meta-heuristic and evolutionary algorithms have been used in this field.

The particle swarm optimization (PSO) algorithm is a classic and important method in solving problems of continuous nature. In [11], and using the PSO algorithm, a multi-objective algorithm has been introduced for server placement with the objectives of improving energy consumption and more effective use of resources. In this algorithm, each server is considered a particle. The authors improved and adapted it for server placement that has a discrete nature. Also, in [12], using PSO and genetic algorithm(GA), a new method of edge server placement has been introduced with the objectives of energy consumption optimization and better load balancing of servers. GA and PSO have been used in service offloading and optimization of optimal server placement strategy, respectively. PSO algorithm has been used in another study [13]. In this paper, the PSO is used to assign servers to locations in the 5G network. This algorithm uses local and global experience in choosing the optimal location of servers. Consequently, this strategy helps to find the best locations for servers. The objectives of this paper are to reduce both the energy consumption of resources and server access delay. At the beginning of the algorithm, servers are randomly

placed in candidate locations, and then their optimal placement is gradually optimized by the iteration of the PSO algorithm.

GA as an evolutionary algorithm has been used in many papers. This algorithm also has been deployed in some server placement methods. In [14], using GA, hill climbing (HC), and simulated annealing (SA) algorithms, a new method of server placement has been introduced. In this article, each server is considered as a Gene, and a solution that includes the number of servers is modeled as a Chromosome. SA and HC algorithms are used to further exploration of the environment to find new solutions and determine a state between a server location and server connection, respectively. Reducing latency and better load balancing of servers are the objectives of this paper. In [15] and by using NSGA-ii multi-objective genetic algorithm and decision tree, a method of server placement has been introduced with the objectives of reducing the cost and energy of resources and also improving latency in agricultural automation. Reducing the delay in sending data from agricultural sensors to servers is the main objective of this research. Another model of the multi-objective genetic algorithm, namely NSGA-iii, has been used in server placement [16]. In this research, which was conducted in the field of social media services in the industrial cognitive internet of vehicles, the CQP algorithm has been used with the objectives of reducing latency and increasing reliability.

Clustering is another technique that has been used in some related research in the field of server placement. For example, in [17], using this technique, a method of server placement has been introduced, with the objective of reducing both communication delay and the number of servers in the 5G network. The basis of the proposed method is based on the reduction of clusters and resources based on a capacitated clustering problem. Also, some heuristics have been used to face the challenges in the algorithm. In another work in this category [18], and using a clustering method, a method of server placement has been introduced. In this algorithm, and in the first step, the resource deployment area is divided into a number of clusters to discover locations with high demands. Then, in the second phase, the processes belonging to each cluster are distributed on the most suitable resources. Finally, in the third phase, the final location of all servers is determined. More effective use of resources is the main goal of this paper. Also, in [19], the authors presented a new clustering-based cloud resource placement model for reducing task overhead. The basis of this method is an adaptive clustering of network access points with the objective of reducing both energy consumption and response time. Then tasks are distributed on the servers based on their processing capacity.

Some other techniques used in this field are integer programming based [20], learning based [21], game theory based [22], Dynamic Programming based [23], and Approximation Algorithm based [24].

A review of related papers shows that despite some solutions to the server placement problem, there are still some challenges. Some methods that are based on population and iteration have difficulties, like converging and reaching the optimal solution, as the problem gets bigger. In the methods based on clustering, finding the optimal number of clusters and avoiding falling into the local optima are considered some challenges in this field. In learning-based methods, the number of inputs cannot exceed a certain limit, and there will be a possibility of overtraining and increasing time complexity. The proposed method of this paper, using BOA and CRO algorithms, is designed to overcome the challenges in edge server placement problems.

3. Research foundations

In this section, the basics of the proposed method, including BOA and CRO algorithms, as well as the DVFS technique, have been described.

3.1 BOA

BOA [8] is a meta-heuristic algorithm based on the behavior of butterflies in sensing and processing the smell of flowers. The quality of each flower or modality is determined based on three parameters: sensory modality (C), stimulus intensity (I), and power exponent (a). sensory modality means determining the quality of flower fragrance based on inputs. Stimulus intensity is the density or amount of the fragrance, which is considered the quantity of a solution in the BOA algorithm. And finally, the Power exponent is a parameter for strengthening and non-linearly behavior of the solution. Equation (1) shows the fitness calculation of a solution that is measured according to the value of the fragrance, which means how attractive this fragrance is to other butterflies. In most cases, a and c are considered random numbers between [0,1]. When a = 1, there is no absorption of fragrance or the amount of fragrance emitted by a particular butterfly is sensed in the same capacity by the other butterflies. Also, a=0 means that the emitted fragrance is not sensed by any other butterfly. Therefore, the value of this parameter controls the behavior of the algorithm. Finally, parameter C determines the speed of convergence.

$$f = CI^a \quad (1)$$

BOA is an iterative-based method that has two types of global and local searches. Equation (2) shows the global search of BOA. In this search, a butterfly randomly moves to the best solution or g^* to obtain its new position.

$$x_i^{t+1} = x_i^t + (r^2 \times g^* - x_i^t) \times f_i \quad (2)$$

where x_i^t is equal to solution x_i for the ith butterfly in iteration number *t*. g^* is the best current global solution among all butterflies. f_i is also the amount of fragrance of butterfly i. Finally, *r* is a random number between [0,1].

Equation (3) shows the local search model of the BOA algorithm. where x_j^t and x_k^t are the jth and kth butterflies in the local search area, respectively, and the current solution will move toward the area between them. If x_j^t and x_k^t belong to the same swarm, and *r* is a random number between [0,1], then equation 3 becomes a random search.

$$x_i^{t+1} = x_i^t + (r^2 \times x_j^t - x_k^t) \times f_i \quad (3)$$

3.2 CRO

The coral reefs optimization [9] is an evolutionary algorithm based on the life of corals on reefs. This algorithm has been used in some research related to traditional cloud computing [25,26]. In this algorithm, the two-dimensional model of the system is represented by $\Lambda = N * M$. Each member $(i,j) \in \Lambda$ of this two-dimensional network is a coral, i.e., $\Xi_{i,j}$, which is coded with the alphabet $\mathcal{A}$ and represents different solutions to a problem. The ratio of empty to occupied cells is called ρ_0, where $0 < \rho_0 < 1$. The fitness of a solution or coral is determined by the health function, where $f(\Xi_{i,j}): \mathcal{A} \to \mathbb{R}$. The various operators of this algorithm include the following:

(1) *Broadcast Spawning*(*External Sexual Reproduction*).

In each initial stage of reef formation, F_b percent of corals are randomly selected for external sexual production, and the recombination process is performed on them. The remaining corals $(1 - F_b)$ will be used later. New larvae are generated, and then their fitness is determined.

(2) *Brooding*(*Internal Sexual Reproduction*)

$1 - F_b$ percentage of the remaining corals of the previous stage participate in a mutation-like operation. This stage is called internal sexual reproduction. The new larvae are added to the population to participate in the competition to settle on the reefs.

(3) *Larva Setting*

At this stage, the larvae randomly try to choose the holes on the reef. If the holes are not empty, they can replace the previous coral only when they have more merit than them. If the replacement process fails after several attempts, these corals will be terminated.

(4) *Budding or fragmentation*(*asexual reproduction*)

In this step, the fitness of all corals settled on a reef is determined by the function $f(\Xi_{i,j}): \mathcal{A} \to \mathbb{R}$, and are sorted according to their fitness, then F_a percent of the best ones reproduce themselves to give more chance to good solutions.

(5) *Depredation*

In each stage, F_d percentage of weak corals die. This will increase the number of empty holes for the establishment of next corals and increase the exploration capability.

3.3 Energy model and DVFS Technic

The energy consumption of a server's processor includes its static and dynamic energy. The static energy of a processor is related to its design and hardware structure. The dynamic energy of a processor is related to its working voltage and frequency. Equation (4) shows the energy consumption of a processor, where E is equal to total energy and E_s and E_d are static and dynamic energy, respectively.

$$E = E_s + E_d \qquad (4)$$

The dynamic energy is shown in Equation (5). where v ،f ،P_d ،t_p are the voltage, frequency, power, and time interval of processor activity, respectively. k is also constant and related to processor technology.

$$E_d = \sum_{t_p}(P_d \times t_p) = \sum_{t_p}(k \times v^2 \times f \times t_p) \qquad (5)$$

Static power is shown in Equation (6) where $v_{j.min}$ ،$f_{j.min}$ ،$t_{j.idle}$ are the minimum voltage, minimum frequency, and activity time in processor j in idle mode, respectively [26-28].

$$E_s = E_{idle} = \sum_{j=1}^{n}(k_j \times v_{j.min} \times f_{j.min} \times t_{j.idle}) \qquad (6)$$

One of the important features of modern processors is having different power modes. This technique is called DVFS (dynamic voltage and frequency scaling). Table 1 shows different voltage and frequency specifications of AMD Turion MT-34 and AMD Opteron 2218 processors with different power modes[29]. This technique ensures using the lowest power mode of the processor to execute a process, taking into account task requirements and its expiration time. This technic significantly reduces energy consumption.

Table 1. The power settings of AMD processors[29]

AMD Turion MT-34			AMD Opteron 2218		
Frequency (GHz)	Voltage (V)	Power (W)	Frequency (GHz)	Voltage (V)	Power (W)
0.8	0.90	6.25	1.0	1.10	26.16
1.0	1.00	9.65	1.8	1.15	51.47
1.2	1.05	12.76	2.0	1.15	57.19
1.4	1.01	16.34	2.2	1.20	68.49
1.6	1.15	20.41	2.4	1.25	81.08
1.8	1.20	25.00	2.6	1.30	95.00

3.4 Latency model

Latency refers to the time difference between the server response time and a user service request time. This time must be tolerable, especially in online applications. Otherwise, such services will not be usable. Latency is a function of the distance between each CBS and MES, as well as the Bandwidth of the communication network and the volume of

data sent or received. In most of the research, this time is only considered proportional to the distance between a CBS and MES, without considering the network traffic and the Bandwidth of the communication network. In the proposed method of this paper, latency is computed using Equation (7).

$$Latency(S,C) = Pd(S,C) + Td(S,C) \quad (7)$$

Where Pd and Td are equal to propagation delay and transmission delay, respectively. The propagation delay of a signal depends on the length of the transmission line between Server S and the cellular base station C and the propagation speed, which is shown in Equation (8). The distance between a CBS and MES is measured using Euclidean distance. The propagation speed also depends on the propagation environment, which is proportional to the speed of light.

$$Pd = \frac{D(S,C)}{Speed} = \frac{\sqrt{(x_C - x_S)^2 + (y_C - y_S)^2}}{speed} \quad (8)$$

The delay in sending a packet is also related to the size of the sent packet and the Bandwidth of the transmission medium, which is shown in Equation (9).

$$Td = \frac{Data-size}{BW} \quad (9)$$

Where $Data - size$ and BW are equal to the size of the sent or received data and the Bandwidth of the communication line, respectively.

The symbols used in this article and their explanations are given in Table 2

Table 2. symbols and descriptions

Symbol	Description
MES	Mobile Edge Server
CBS	Cellular Base Station
DVFS	Dynamic Voltage and Frequency Scaling
E_s	Static Energy
E_d	Dynamic Energy
E	Total Energy
CRO	Coral Reefs Optimization Algorithm
BOA	Butterfly Optimization Algorithm
pop_{size}	Population Size
itr_{num}	Iteration number
Pd	Propagation delay
Td	Transmission delay

4. Proposed method

In this section, the details of the proposed model are stated. The main entities of the proposed method are the set of base stations and servers, where $B = b_1, b_2 \dots b_n$ and $S = s_1, s_2 \dots s_m$ are the sets of n base stations and m servers, respectively. Each server belongs to one or more base stations that is in their range. Mobile network users can access resources via cellular base stations. Considering the objectives of the proposed method and the tradeoff between them, i.e., reducing both energy consumption and resource access delay, we define a new normalized objective function model using the weighted average sum method by Equation (10).

$$F = \min\left(w_1 \times \text{norm}\frac{\sum_{i\in B, J\in S} latency(i,j)}{|E|} + w_2 \times \text{norm}\frac{\sum_{k\in S} E_k}{m}\right) \quad (10)$$

where $latency(i,j)$ is the delay between server i and base station j, $|E|$ is The total number of network links that connect the servers to the base stations, E_k is the energy consumption of the server k, and m is the total number of resources. w_1 and w_2 are the weights of the objectives which can be adjusted. Norm is the normalized value of each objective. In the proposed method, each mapping between the servers and the base station is considered a solution. Due to a large number of servers and CBSs and to avoid increasing the time complexity of finding the solution, the

problem space is divided into smaller zones. In local search, the location server in each area is determined using the BOA algorithm, and in global search, the CRO algorithm will be used for maping between servers and locations. Each server is covered by one or more CBSs, and each CBS has access to only one server. The location of each server among its covered CBSs plays an important role in server access delay reduction. Figure 1 shows the network model.

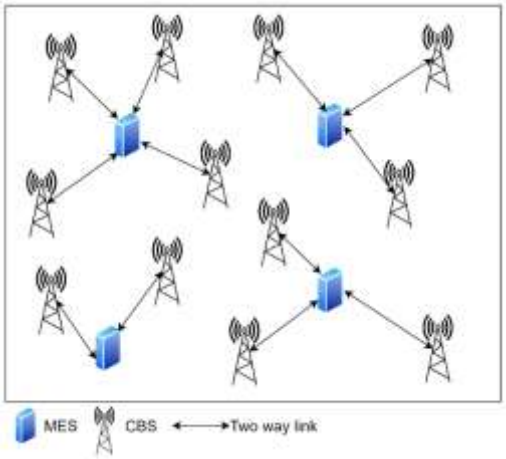

Fig.1 Network model

In this model, each MES is covered by one or more CBSs. Considering the heterogeneous nature of the service requests in different parts of the network, finding the optimal resource locations and finding the best server, taking into account the objectives of the proposed method, is the main goal of the proposed method. In each area, some locations are randomly selected for server placement. If Z_k is the region k in the network, then $v_{k=}^{i} f(x(t)_k^i$ is equal to the value of the current location or $x(t)$ of ith MES in zone *k* and iteration *t* that is computed by Equation (10). After determining the fitness of all initial solutions, the best location of the server is determined as x_k^*. At this stage, all current locations are updated to be closer to x_k^*. This update is done both globally and locally. Equation (11) shows the global search of the server locations in the proposed method.

$$x(t+1)_k^i = x(t)_k^i + (r^2 \times x_k^* - x(t)_k^i) \times v_k^i \quad (11)$$

Where $x(t+1)_k^i$ is the new location of ith solution *i* in zone *k*, and *r* is the random number between [0,1]. Local search is used to further explore the search space, avoiding greedy search and forcing the algorithm to more random behavior. Equation (12) updates the position of a candidate location, where j and s are two random locations to which the current position randomly moves toward them.

$$x(t+1)_k^i = x(t)_k^i + (r^2 \times x(t)_k^j - x(t)_k^s) \times v_k^i \quad (12)$$

And at the end of this step, the best solutions obtained in each iteration are transferred to the next round of the algorithm. Algorithm 1 shows the details of this step.

Algorithm.1 local server placement
Initialize parameters (pop_{size}, itr_{num}, p, w_i) 1: objective function $f(x), x = (x_1, x_2, \dots x_{dim})$, dim=no of dimensions 2: generate the initial population of *n* Butterflies $x_i = (i = 1,2, \dots n)$ 3: $itr = 0$ 4: **for** each zone 4: **for** each solution *bf* in the population, do 5: compute the fitness of each solution x_i^t by $f(x_i^t)$ using Eq. (10) 6: **end For** 9: find the best *bf* 10: **for** each butterfly *bf* in the population, do

```
11:              generate a random number r from [0, 1]
12:              If r < p then
13:                  move towards best location(x_k^*) using Eq. (11)
14:              else
15:                  move randomly to solutions  using Eq. (12)
16:              end if
17:          end for
18:  end for
19:  itr = itr + 1
20:  if itr < itr_num
21:     continue the algorithm
22:  else:
23:     return the best location for each zone
```

After the first step of the proposed method, i.e., finding the optimal location of the server in each area, it is time for the second step of the proposed method. In the first step, the best location of a randomly selected MES is determined for each region, but there is no guarantee that this server is the best suitable MES needed in that area. In the second step of the proposed method, and by using the CRO algorithm, the best MES that is suitable for each area will be selected. The optimal server placement model of each area is shown in Table 3, where a server s_i is assigned to a region k_i (for $i = 1\ to\ k$).

Table 3. servers to zones assignments

z_1	z_2	z_3	z_4	…	z_k
s_1	s_2	s_3	s_4		s_k

As mentioned before, The model is inspired by the CRO algorithm. In this algorithm, the parameter $0 < \rho_0 < 1$ is the ratio of empty to occupied cells, and the optimal value of this parameter has an important role in the performance of this method. For further explanation, an example with nine regions and 12 servers is considered, which is shown in Table 4. Obviously, three servers will not be used in every solution vector. Therefore, the ratio of unused servers to used servers is determined as 3 to 9 or $\rho_0 = 0.33$. Unused servers are different in each solution vector. Calculating the optimal value of ρ_0 plays an important role in improving the performance of the proposed method. Its increase makes the problem space more complicated, whereas its decrease causes early convergence.

In the recombination phase (*Broadcast Spawning*), the new solution vector will be obtained in a multi-parent sequential manner [30], as shown in Table 4. The results of the experiments performed on this recombination method have shown that it has better efficiency than the simple order recombination method. Based on this method, four solutions($sv_1\ to\ sv_4$) are combined in an orderly manner and create a new solution (nsv_5). The first parent(sv_1) has the best fit as the base of the recombination. With this technique, good solutions have more chances to participate in the recombination process.

Table 4. Broadcast spawning operator

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9
sv_1	s_3	s_{10}	s_9	s_2	s_1	s_{12}	s_4	s_6	s_7
sv_2	s_9	s_1	s_{10}	s_7	s_3	s_4	s_6	s_2	s_{12}
sv_3	s_{10}	s_6	s_3	s_1	s_7	s_2	s_9	s_4	s_{12}
sv_4	s_7	s_2	s_9	s_4	s_3	s_1	s_{10}	s_{12}	s_6

nsv_5	s_7	s_3	s_9	s_2	s_1	s_4	s_6	s_{10}	s_{12}

The second operator of this step is *Brooding*, in which the new solution is generated by changes of only one parent. In the proposed method, swapping mutation is performed over the servers belonging to the regions. In this case, a number of servers are randomly exchanged together. Table 5 shows an example of this operator. In this example, servers s_3 and s_{10} of solution vector sv_1 are randomly selected and exchanged with each other and create a new solution vector nsv_2.

Table 4. Brooding operator

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9
sv_1	s_7	$\boldsymbol{s_3}$	s_9	s_2	s_1	s_4	s_6	$\boldsymbol{s_{10}}$	s_{12}

nsv_2	s_7	$\boldsymbol{s_{10}}$	s_9	s_2	s_1	s_4	s_6	$\boldsymbol{s_3}$	s_{12}

In order to preserve the good solutions, a number of the best solutions should be duplicated. Therefore, F_a percentage of the best ones is calculated by using equation 10 to increase the chances of finding better solutions in the current iteration, which is called *budding*. And finally, to increase the exploration capability of the proposed method, the p_d percentage of weak solutions is replaced by random solutions. This phase is called *Depradation*.

At the end, all the generated solutions are sorted according to their fitness and some of the best ones are selected for the next iteration. The population number is pop_{size} and the number of repetitions of the algorithm is itr_{num}. Algorithm 2 shows the global placement of resources in the proposed method.

Algorithm.2 global server placement

Input: best servers location of k regions
Output: best servers for all k regions

1: initialize parameters(pop_{size} , itr_{num} , F_b, F_a, p_d, ρ_0 , k)
2: create pop_{size} random solution vectors with respect to ρ_0
3: itr=0
4: use the lowest possible power mode for servers
5: apply broadcast spawning on F_b percent of solution vectors
6: apply Brooding on $1 - F_b$ percent of solution vectors
7: apply Budding on F_a percent solution vectors
8: compute fitness of all solution vectors using Eq. (10)
9: add all new solution vectors to the population
10: if random(0, 1)< p_d
11: no operation!
12: else
13: apply Depredation on F_d percent of worst solution vectors
14: eliminate pop_{size} of best solution vectors and add them to the population
15: $itr = itr + 1$
16: if $itr < itr_{num}$
17: continue the process
18: else
19: return the best solution vector of the population

5. Evaluation and simulation results

In this section, the proposed method is evaluated, and test scenarios, compared algorithms, and simulation environment are described. Also, the performance of the proposed method is evaluated in various experiments.

5.1 Experiments Setup and Dataset

To evaluate the proposed method, the cellular communication network of Tehran has been used. As the capital of Iran, Tehran is the most important political and economic city, with 750 square kilometers and a population of about 9 million people. This city has flat, mountainous, and marginal areas, and the demands for services during different times and areas are not equal. Most of the administrative, educational and commercial areas are located in the center of the city, and the population density varies in different areas. Figure 2 [31] shows a part of the cellular telecommunication map of the MCI operator [32] of the 4G mobile network in Tehran. The numbers inside the circles indicate the number of antennas covered in that area. MATLAB software version R-2016a has been used on a computer with Intel (R) Core (TM) i7 2.50 GHz processor, 64-bit operating system, and 16.0 GB Ram for simulation. Table 5 shows a part of the geographic characteristics of CBS antennas in Tehran, which include cell ID (CID), longitudinal and latitude coordinates, covered range, and the average number of users on each base station. Also, server features used in three different types and different processing power are given in Table 6, which include frequency, voltage, and static power of servers. Data packets in size of 1 to 4 Giga Bytes are generated and sent based on the number of users of each CBS, with Poisson distribution with $\lambda = 4$ per minute. The Bandwidth of the communication network is considered 1 Giga Bytes per second.

Fig. 2. Part of MCI cellular base stations of Tehran

Table 5. The CBSs information on Tehran City

Row	CID	Latitude	Longitude	Range	Online users
1	4675730	35.689773	51.390609	1000m	221
2	4675733	35.690048	51.389786	1000m	563
3	4668661	35.689912	51.392396	1000m	233
4	4677604	35.689087	51.393356	1000m	913
5	4724596	35.687946	51.391856	1000m	301
…	…	…	…	…	…
320	4743140	35.687714	51.391983	1000m	51

Table 6. Servers features

Type	Frequency (GHz)	Voltage (V)	Power (W)
A	2.0	2.2	30
	2.5	2.7	40
	3.0	3.5	50
B	4.0	5	75
	5.0	7.8	100

	6.5	11	120
C	8.0	15	150
	10.0	21	175
	12.0	32	220

5.2 Compared algorithms

To evaluate the proposed method, three related and state-of-the-art algorithms have been used. In [11], a PSO-based multi-objective server placement algorithm with the objectives of server energy reduction and more efficient use of resources has been introduced as similar work on the Shanghai Telecom telecommunication network. Based on this algorithm, each solution is considered a particle, and for simplicity, it has been assumed that all servers are homogeneous. Due to the feature of the PSO algorithm that is suitable for continuous problems, the authors have made changes to this algorithm to design a new model with the ability of server placement that has a discrete nature. in [5], a learning-based server placement algorithm that uses the Deep Q-Network model, and CRO algorithm (MOP-DQ) has been introduced. To reduce the time complexity of the server placement problem, the authors clustered the resource deployment area into small sub-regions. CRO and Deep Q-Network algorithms are used for local and global search processes, respectively. Latency reduction and better load balancing are the objectives of this algorithm. Finally, in [12], and using genetic and PSO algorithms, a new server placement method EPMOSO has been introduced with the objectives of better load balancing of servers, reducing the energy consumption of resources, and reducing the resource access delay. This algorithm is based on service offloading on the Internet of Things. Genetic and PSO algorithms have been used in the field of service offloading and optimization of optimal server placement strategy, respectively.

5.3 Evaluation results

In this section, the proposed method is evaluated and compared with similar algorithms. Optimization of servers' energy consumption, servers' average access delay reduction, and also reducing the number of used resources are the objectives of the proposed method.

A: Energy consumption

Reducing server energy consumption not only saves resource utilization costs but also avoids the production of toxic pollutants. Figure 3 shows the energy consumption of the proposed method and other algorithms for 50 to 175 different servers while the number of BTSs is constant. As this figure shows, the proposed method has better efficiency in energy consumption compared to other algorithms. The reason for this superiority is due to the following reasons.

1. Applying continuous features and accurate local and global search of BOA algorithm to find servers deployment locations by using its powerful operators.
2. Using the coral reef optimization algorithm, which has a suitable convergence speed in solving discrete optimization problems, to find the best server in the locations determined in the first step of the proposed method.
3. Using the DVFS technique to operate MESs at their minimum power mode.

As this figure shows, when the number of servers increases, the performance of the proposed method improves compared to other methods.

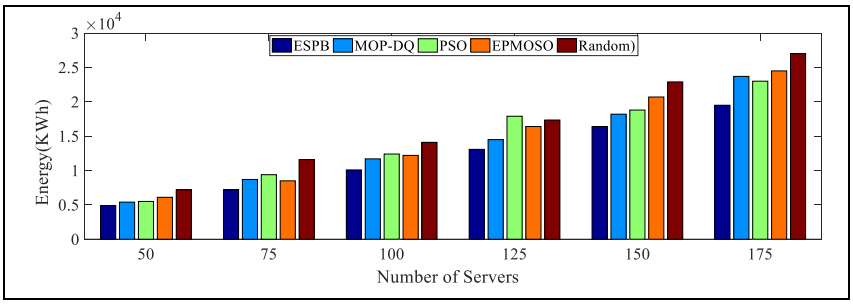

Fig. 3. Energy consumption of the servers(320 CBS)

Although reducing the number of CBSs will reduce the overall energy consumption of the network, it may cause the network can not to meet the needs of all users. This forces the network to use servers located in traditional cloud data centers to respond to some requests. Consequently, due to the long distance between cloud data centers and mobile network users, some services are received with delay, which cannot be tolerated in some cases, especially in online applications. This metric is called the local network server access error. Increasing the number of resources reduces this error. On the other hand, excessive use of resources will reduce network efficiency. Figure 4 shows the local network server access error rate in the proposed method and the compared algorithms. As this figure shows, the error rate of the proposed method is lower than other algorithms. The reason is the proper performance of the proposed method in better placing network edge servers.

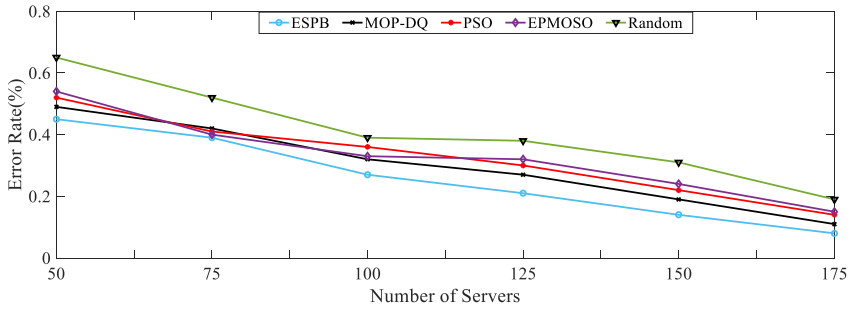

Fig. 4. local network server access error rate

In the next scenario, we keep the number of servers constant and change the number of CBSs. The energy consumption of the resources is measured simultaneously. By keeping the number of servers constant, the increase of CBS will lead to the coverage of more users in the network, and the availability of servers will increase. However, the optimal placement of resources will play an important role in reducing the error rate and optimal energy consumption. Figure 5 shows this problem. In this figure, the number of servers is 100, and the number of CBS will vary between 200 and 320. It is obvious that reducing the number of CBSs leads to the reduction of the coverage area. As Figure 5 shows, due to the performance of the proposed method, the energy consumption of servers has decreased significantly compared to other methods. The reason is the better placement of servers and deploying the most suitable server in each region.

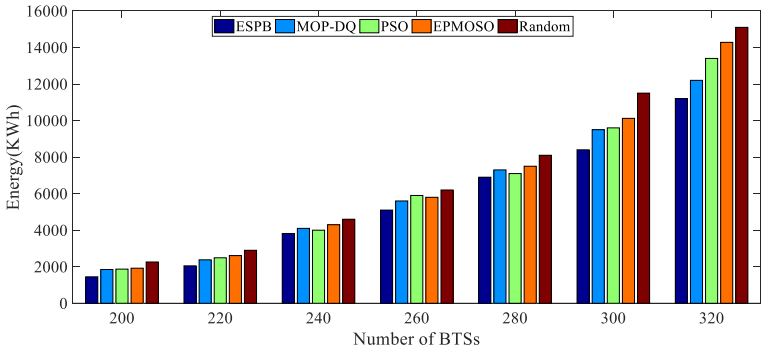

Fig. 5 Energy consumption of the servers(100 MES)

B: Latency

Latency is the time difference between the time of sending the request and the response of the server. This time is a function of the distance between the user and the server, the type of transmission medium, and the Bandwidth of the communication network. In most research, this delay is only considered proportional to the distance between a CBS and MES. The proposed method, besides this parameter, will try to reduce this time by considering all related parameters. Finding the optimal location and placing the most suitable server helps to reduce the average network latency. The proposed method, by using the BOA algorithm, which is suitable for problems of continuous nature, instead of choosing predefined candidate locations, tries to search the whole area and find the optimal location of resources. After finding the best resource deployment locations, it is time to choose the most suitable server for each region. in this step CRO algorithm finds the best mapping between resources and locations. In some studies [7], and for simplicity, it is assumed that the resources have the qual processing power and are homogeneous. By removing this limitation and paying attention to the real conditions of the network, the proposed method seeks to find the most suitable servers according to the needs of a region. Figure 6 shows the latency of the proposed method and compared algorithms. As this figure shows, the proposed method has less latency than other methods. The reason for this improvement is due to consider all parameters affecting this delay and also to search the whole space to find the optimal location of servers.

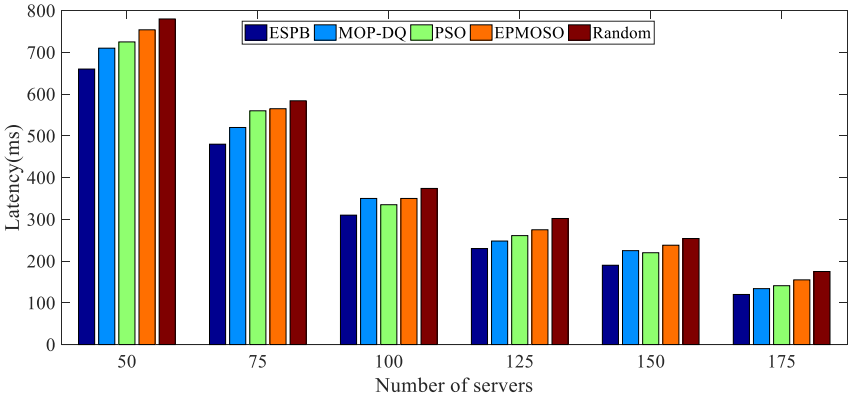

Fig. 6 Average latency of the network(320 CBS)

Another scenario for calculating latency is to keep the number of servers constant and change the number of CBSs. It is expected that with the reduction of CBSs, the network coverage area will decrease, and some requests will face errors. In this case, the network servers cannot respond to all requests due to the low coverage, and consequently, some requests are sent to the cloud data center, which can increase the latency. However, this delay in the proposed method is less compared to other algorithms. The reason for this reduction is due to better server placement, which will result in fewer errors. As mentioned earlier, the proposed method places the cloud servers without limitation in the selection of candidate locations. It also finds the optimal locations and assigns the most suitable resource to each area. Figure 7 shows the average network latency in this scenario.

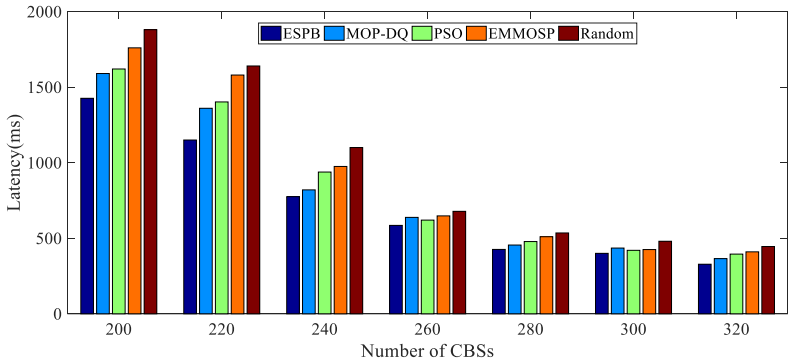

Fig. 7 Average latency of the network(100 MES)

C: Performance metrics

In this section, the efficiency of the proposed method is measured by two criteria. The first metric is the average network resource utilization rate, and the second one is the minimum number of servers to be used without reducing network efficiency. The appropriate server placement causes that due to the volume of users' demands, the most suitable server is assigned to that area. Otherwise, there will be a possibility of servers overloading while the servers of the adjacent areas are underloaded. This increases latency, reduces reliability, and increases energy consumption. Our proposed method, in its first step, finds the best server location of each area, without any restrictions on the location of resources, and then assigns the most suitable resource to that area in the second phase. Table 7 shows the average server utilization. As expected, the proposed method, in most cases, has a better utilization rate except in the first column, where the number of servers is 50. It is obvious that reducing the average resource utilization is directly related to reducing the number of servers. Because, in this case, the network local server access error increases.

Table 7. Average resource utilization

Algorithm	Number of edge servers					
	50	75	100	125	150	175
ESPB	0.55	**0.62**	**0.69**	**0.75**	**0.84**	**0.93**
MOP-DQ	**0.57**	0.59	0.62	0.66	0.69	0.75
PSO	0.52	0.54	0.58	0.62	0.68	0.75
EPMOSO	0.49	0.52	0.55	0.59	0.63	0.69
Random	0.40	0.43	0.46	0.50	0.54	0.58

The second efficiency metric of the proposed method is the number of used servers. In other words, the minimum number of servers that still maintain the efficiency of a method is considered. Reducing the number of servers will reduce both energy consumption and cost. Table 8 shows the efficiency of algorithms proportional to reducing the number of servers. The performance of each method has been measured independently and relative to the objective function of that method. According to Table 8, by using 175 servers, each method performed best. By reducing the number of servers to 170, the proposed method still maintains its efficiency, but the efficiency of other methods is reduced. When the number of servers is reduced to 165, the performance of all methods, including the proposed method, decreases, but this decrease for the proposed method is at a lower rate. As a result, the proposed method still continues to work without performance reduction with 170 servers, while the efficiency of the other methods decreases gradually. This saving in the number of servers will play an important role in reducing the cost and energy of resources.

Table 8. performance of the algorithms

Algorithm	Number of edge servers					
	175	173	170	165	160	155
ESPB	1.00	1.00	**1.00**	0.97	0.93	0.9
MOP-DQ	**1.00**	0.98	0.96	0.92	0.88	0.81
PSO	**1.00**	0.97	0.95	0.93	0.89	0.82
EPMOSO	**1.00**	0.95	0.94	0.91	0.87	0.84
Random	**1.00**	0.93	0.90	0.87	0.83	0.79

6. Conclusion

effective server placement plays an important role in providing appropriate services to mobile users and also saves resources. Users prefer to receive services with low latency, while cloud service providers want to reduce both energy consumption and costs. For this reason, in this article, using BOA and CRO algorithms, a new two-step multi-objective server placement method has been introduced. Using the BOA algorithm, which is suitable for solving continuous optimization problems, makes it possible to determine the location of a server in an area. Then, by using the CRO algorithm, the best mapping between servers and their deployment locations is obtained. Also, the DVFS technique makes servers work at minimum power mode to save energy. as a result, the proposed method reduces the energy consumption of servers and reduces both the network latency and the number of used servers. For future work suggestions, first, we intend to use the proposed method in other fields, such as offloading and workload balancing on resources. Also, as a second suggestion for the development of the proposed method, we plan to use parallelization techniques to increase its efficiency and speed.

Declarations

Ethical Approval: Not applicable

Competing interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors' contributions: Asghari and sayadi wrote the main manuscript. Azgomi wrote the program code. sayadi reviewed the manuscript. Asghari translated the manuscript into English.

Funding: No Funding

Availability of data and materials: The datasets generated during the current study are available from the corresponding author upon request

References

[1] Asghari, Ali, Mohammad Karim Sohrabi, and Farzin Yaghmaee. "Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm." *The Journal of Supercomputing* 77, no. 3 (2021): 2800-2828.

[2] Chang, V., 2018. An overview, examples, and impacts offered by emerging services and analytics in cloud computing virtual reality. *Neural Computing and Applications*, *29*(5), pp.1243-1256.

[3] Lin, H., Zeadally, S., Chen, Z., Labiod, H. and Wang, L., 2020. A survey on computation offloading modeling for edge computing. *Journal of Network and Computer Applications*, *169*, p.102781.

[4] Fernando, N., Loke, S.W. and Rahayu, W., 2013. Mobile cloud computing: A survey. *Future generation computer systems*, *29*(1), pp.84-106.

[5] Asghari, A. and Sohrabi, M.K., 2022. Multi-objective edge server placement in mobile edge computing using a combination of multi-agent deep Q-network and coral reefs optimization. *IEEE Internet of Things Journal*.

[6] Asghari, A. and Sohrabi, M.K., 2022. Bi-objective cloud resource management for dependent tasks using Q-learning and NSGA-3. *Journal of Ambient Intelligence and Humanized Computing*, pp.1-21.

[7] Wang, S., Zhao, Y., Xu, J., Yuan, J. and Hsu, C.H., 2019. Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, *127*, pp.160-168.

[8] Arora, S. and Singh, S., 2019. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Computing*, *23*(3), pp.715-734.

[9] Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-López, S. and Portilla-Figueras, J.A., 2014. The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. *The Scientific World Journal*, *2014*.

[10] Wu, C.M., Chang, R.S. and Chan, H.Y., 2014. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems*, *37*, pp.141-147.

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

[12] Ma, R., 2021. Edge Server Placement for Service Offloading in Internet of Things. *Security and Communication Networks*, *2021*.

[13] Li, Y., Zhou, A., Ma, X. and Wang, S., 2021. Profit-aware edge server placement. *IEEE Internet of Things Journal*, *9*(1), pp.55-67.

[14] Kasi, S.K., Kasi, M.K., Ali, K., Raza, M., Afzal, H., Lasebae, A., Naeem, B., Ul Islam, S. and Rodrigues, J.J., 2020. Heuristic edge server placement in industrial internet of things and cellular networks. *IEEE Internet of Things Journal*, *8*(13), pp.10308-10317.
[15] Zhang, J., Li, X., Zhang, X., Xue, Y., Srivastava, G. and Dou, W., 2021. Service offloading oriented edge server placement in smart farming. *Software: Practice and Experience*, *51*(12), pp.2540-2557.
[16] Xu, X., Shen, B., Yin, X., Khosravi, M.R., Wu, H., Qi, L. and Wan, S., 2020. Edge server quantification and placement for offloading social media services in industrial cognitive IoV. *IEEE Transactions on Industrial Informatics*, *17*(4), pp.2910-2918.
[17] Lee, S., Lee, S. and Shin, M.K., 2019, October. Low cost MEC server placement and association in 5G networks. In *2019 International conference on information and communication technology convergence (ICTC)* (pp. 879-882). IEEE.
[18] Mohan, N., Zavodovski, A., Zhou, P. and Kangasharju, J., 2018, August. Anveshak: Placing edge servers in the wild. In *Proceedings of the 2018 Workshop on Mobile Edge Communications* (pp. 7-12).
[19] Li, B., Hou, P., Wu, H., Qian, R. and Ding, H., 2021. Placement of edge server based on task overhead in mobile edge computing environment. *Transactions on Emerging Telecommunications Technologies*, *32*(9), p.e4196.
[20] Cui, G., He, Q., Chen, F., Jin, H. and Yang, Y., 2020. Trading off between user coverage and network robustness for edge server placement. *IEEE Transactions on Cloud Computing*.
[21] Kasi, M.K., Abu Ghazalah, S., Akram, R.N. and Sauveron, D., 2021. Secure mobile edge server placement using multi-agent reinforcement learning. *Electronics*, *10*(17), p.2098.
[22] Cao, K., Li, L., Cui, Y., Wei, T. and Hu, S., 2020. Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. *IEEE Transactions on Industrial Informatics*, *17*(1), pp.494-503.
[23] Wang, F., Huang, X., Nian, H., He, Q., Yang, Y. and Zhang, C., 2019, December. Cost-effective edge server placement in edge computing. In Proceedings of the 2019 5th international conference on systems, control and Communications (pp. 6-10).
[24] Meng, J., Shi, W., Tan, H. and Li, X., 2017, August. Cloudlet placement and minimum-delay routing in cloudlet computing. In *2017 3rd international conference on big data computing and communications (BIGCOM)* (pp. 297-304). IEEE.
[25] Asghari, A. and Sohrabi, M.K., 2021. Combined use of coral reefs optimization and reinforcement learning for improving resource utilization and load balancing in cloud environments. *Computing*, *103*(7), pp.1545-1567.
[26] Asghari, A. and Sohrabi, M.K., 2022. Combined use of coral reefs optimization and multi-agent deep Q-network for energy-aware resource provisioning in cloud data centers using DVFS technique. *Cluster Computing*, *25*(1), pp.119-140.
[27] M. Safari, R.Khorsand, Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment, *Simulation Modelling Practice and Theory*, 87 (2018) 311-326.
[28] Shirvani, Mirsaeid Hosseini, Amir Masoud Rahmani, and Amir Sahafi. "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges." Journal of King Saud University-Computer and Information Sciences 32, no. 3 (2020): 267-286.
[29] [26] Shirvani, Mirsaeid Hosseini, Amir Masoud Rahmani, and Amir Sahafi. "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges." Journal of King Saud University-Computer and Information Sciences 32, no. 3 (2020): 267-286.
[30] Arram, A. and Ayob, M., 2019. A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems. *Computers & Industrial Engineering*, *133*, pp.267-274.
[31] https://www.cellmapper.net/map: Feb 10, 2022
[32] https://mci.ir/