

User Experience Oriented Edge Server Placement

Shuliang Wang (✉ Slwang2011@bit.edu.cn)

Beijing Institute of Technology

Shuo Liu

Beijing Institute of Technology

Yijie Yang

Beijing Institute of Technology

Hanning Yuan

Beijing Institute of Technology

Sijie Ruan

Beijing Institute of Technology

Yun Yang

Swinburne University of Technology

Research Article

Keywords: Edge Cloud Computing, Edge Server Placement, Average Utility, Network Robustness, Integer Programming, Heuristic Algorithm.

Posted Date: August 30th, 2023

DOI: <https://doi.org/10.21203/rs.3.rs-3296537/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

User Experience Oriented Edge Server Placement

Shuliang Wang, Shuo Liu, Yijie Yang, Hanning Yuan, Sijie Ruan and Yun Yang,

Abstract—Edge computing holds promising prospects and addresses certain limitations of cloud computing. In the realm of edge computing, service providers deploy edge servers in close proximity to users to minimize network latency. To optimize the utilization of limited edge servers, service providers typically adopt a decentralized approach to their placement. However, in real-world edge computing environments, runtime failures are bound to occur after the placement of edge servers, significantly degrading the user experience. Therefore, edge server placement strategies must consider scenarios involving edge server failures. In general, redundant placement techniques are employed to ensure the reliability of services deployed on edge servers. In practice, edge servers consist of one or more physical machines, which leads us to refer to the physical edge server placement as the PESP problem. From the perspective of mobile users, the PESP solution aims to provide cost-effective and stable services. We measure service stability using a robustness metric and, for the first time, introduce the average utility indicator to gauge service efficiency. Average utility pertains to the performance users obtain from edge servers, such as CPU performance. When placing edge servers, we take into account both average utility and robustness to ensure an optimal user experience. In this paper, we formally model the problem of user experience-oriented physical edge server placement (PESPU) and prove its NP-hardness. To tackle this problem, we propose an optimization method based on integer programming (PESPU-O) to find optimal solutions for small-scale problems. For large-scale problems, we present three heuristics to find solutions. Finally, we conduct experiments using real-world datasets to demonstrate the effectiveness and efficiency of our approaches.

Index Terms—Edge Cloud Computing, Edge Server Placement, Average Utility, Network Robustness, Integer Programming, Heuristic Algorithm.

1 INTRODUCTION

In the past few decades, the number of mobile and Internet of Things (IoT) devices has increased significantly, including mobile phones, portable devices, tablets, and so on. According to the Global System for Mobile Communications (GSMA) report the number of IoT terminal connections worldwide is expected to reach 25 billion by 2025. These mobile devices are expected to have a profound impact on the daily lives of a majority of individuals in the future [1]. However, due to their physical size constraints, these devices suffer from limitations in computing power and battery life [2]. Consequently, running resource-intensive applications such as image recognition [3], natural language processing [4], and interactive games poses significant challenges for these devices [5].

To address the growing computing demands of applications while considering the limited computing capacity of mobile devices, mobile cloud computing has emerged as a solution [6]. By offloading computationally intensive tasks to the cloud, cloud computing overcomes the hardware limitations of user terminals. However, due to significant transmission delays, cloud computing falls short in meeting the stringent latency requirements of certain applications. In order to mitigate this drawback, edge computing has emerged as an alternative [7]. Multi-access edge computing

(MEC) plays a crucial role in the development of 5G and 6G networks [8]. It involves utilizing wireless access networks to provide users with required services and cloud computing functions (edge cloud). Unlike traditional cloud computing, edge computing offers services in close proximity to users, effectively addressing the issue of network delays. Users get service from edge servers [9] on access point or base station nearby and these servers are placed in a distributed fashion [10], [11], [12].

The edge server placement problem (ESP) involves selecting an appropriate geographical location based on a specific strategy while considering constraints related to geographic scope and budget, all while satisfying user requirements. The physical edge server placement problem (PESP) takes into account the fact that an edge server consists of one or more physical machines, and aims to strategically select suitable geographical locations for placing these physical machines. Placing physical edge servers in a MEC environment presents various challenges, as the location of the physical edge server significantly impacts mobile users' access latency and the resource utilization of the server.

Existing research primarily focuses on achieving single objectives, such as performance or network latency, assuming faultless servers that cannot be relocated once placed. However, in real-world scenarios, edge servers are susceptible to factors like abnormal network attacks, hardware failures, and software anomalies, which degrade network robustness. When the initially accessed edge server fails to function properly or the user is out of range of other servers, users are forced to rely on distant cloud services, resulting in a detrimental impact on user experience, particularly during

- S. Wang, S. Liu, Y. Yang, H. Yuan, S. Ruan are with the School of Computer Science and Technology, Beijing Institute of Technology, China. E-mail: Slwang2011@bit.edu.cn; 3220200917@bit.edu.cn; 1120181117@bit.edu.cn; Yhm6@bit.edu.cn; sjruan@bit.edu.cn.
- Y. Yang is with the Department of Computing Technologies, Swinburne University of Technology, Australia. E-mail: yyang@swin.edu.au.

urgent business needs for edge computing services. This issue becomes even more critical in densely populated areas, where a server failure at a base station can prevent a large number of users from accessing services promptly. Consequently, research on the robustness of edge servers holds significant importance. In addition, the user experience is closely tied to the computational capabilities provided by the edge server, which we refer to as utility. However, the utility is strongly influenced by the number of users in the proximity of the edge server within a fixed deployment. Considering these factors, in this study, we define the CPU performance per user as an indicator of utility. Hence, the placement of edge servers should consider both utility and network robustness as crucial factors.

In order to build a MEC environment, we should first formulate appropriate server placement strategies. As an urgent problem to be solved, most of the existing researches are based on the joint optimization of two objectives to find the optimal placement scheme. Most of ESP approaches prompt user utilization as much as possible for a better user experience. This goal is equivalent to minimizing edge server robustness. Edge server placement schemes that maximize network robustness would certainly lead to redundancy in edge server performance. As a result, average utility of users is low. **Considering an actual number of edge servers p , in the problem of physical edge server placement, the user experience-oriented PESP strategy must make an appropriate trade-off between utility and network robustness (PSEPU).**

The main contributions of this paper are as follows:

- 1) We propose the utility index for the first time in the PESP problem to measure the efficiency of user access to services.
- 2) When formulating a PESP strategy for user experience, we first consider the trade-off between utility and network robustness.
- 3) We formally model the PESP problem as a constrained problem and prove that it is NP-hard.
- 4) We propose two solutions to the PESP problem. The first is an optimal solution algorithm based on integer programming, which works on small-scale data sets. The second is three heuristics, which are applied to the feasible solution of the problem on large-scale data sets.
- 5) Extensive experiments are carried out on widely used real-world datasets to evaluate the effectiveness and efficiency of optimal and feasible solutions.

The rest of the paper is structured as follows. We present an example of the motivation for the PESP problem in Section 2. Then, the problem definition, problem formulation, and proof that PESP is NP-hard are given in Section 3. Next, we introduce two approaches to the PESP problem in Section 4, including an integer programming-based approach and heuristics. We evaluate real-world datasets in Section 5 and present related work in Section 6. Finally, we present the conclusions of this paper and propose future work in Section 7.

2 MOTIVATION EXAMPLE

Assuming that an edge server is composed of multiple physical machines, deploying physical machines on the base station can make the placement of edge servers more flexible. As shown in Fig. 1, let us consider simplified scenarios. In this specific case, there exists a solitary base station, and ten users are distributed near the base station. As the number of physical machines escalates, the robustness of the users enhances. However, owing to multi-tenancy, the performance of the cluster diminishes, thereby impacting the average utility. The performance and robustness of the three scenarios are shown in Table 1.

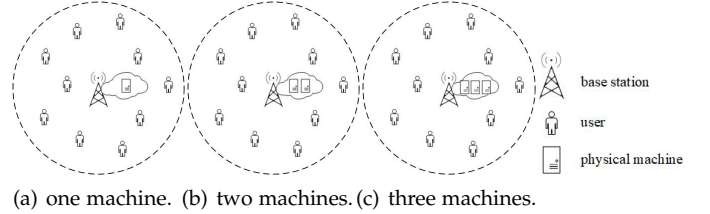


Fig. 1: Simple scenarios.

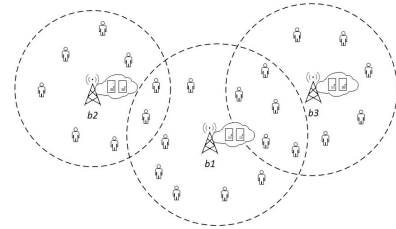


Fig. 2: Complex scenario.

TABLE 1: Performance and robustness of three scenarios.

Scenarios	Performance (CPU utilization)	Robustness
a	best	worst
b	good	good
c	worst	best

Here performance is represented by CPU, and the CPU performance of each physical machine is different. The overall cluster performance is influenced by the number of users in the vicinity and the available computing resources. Robustness, on the other hand, is quantified by assessing the probability of runtime failures in the physical machines. The performance and robustness of scenarios a , b , and c are shown in Table 1.

TABLE 2: Utility and robustness of three scenarios.

Scenarios	Utility	Robustness
(1,4,1)	53.27%	70
(1,3,2)	55.57%	67
(2,2,2)	60.38%	62

In a more complex scenario, there are three base stations, 13 users around base station $b1$, 8 users around $b2$, and 10 users around $b3$. Although the distribution of mobile users is dynamic, we can estimate the approximate number of users

near the base station. Due to budget constraint, only 6 physical machines can be deployed. As shown in Fig. 2, (2, 2, 2) indicates the number of physical machines placed on the base stations b_1 , b_2 , and b_3 . Then all possible placement schemes are as follows: (2, 2, 2), (3, 1, 2), (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 2, 1), (1, 1, 4), (1, 4, 1) and (4, 1, 1).

We measure the robustness by the number of users and the number of physical machines covering them in the region. Utility refers to the performance obtained by users from edge servers, and it is determined by the number of physical machines on the base station and the number of users it covers. Using this measurement method, we can get the robustness and utility values of three different strategies, and they are listed in Table 2. (4, 1, 1) puts more physical machines (4 here) on one base station, so we think it is the same as (1, 4, 1) and (1, 1, 4). These placement strategies place physical machines more centrally so that they have better robustness but poorer utility. On the contrary, such as (2, 2, 2), the physical machines placed on each base station are relatively on average, so the average utility of this placement strategy is the strongest, but the robustness is the worst. In this example, the utility and robustness of (1, 3, 2) are 55.57% and 67. Its utility is better than that of (1, 4, 1), although its robustness is lower. Similarly, its robustness is better than that of (2, 2, 2) and its utility is worse than that. After weighing the utility and network robustness, (1, 3, 2) is a good trade-off based on Table 2 and we take it as the final placement strategy of the scenario shown in Fig. 2. This solution can bring a better user experience, which means it not only has low latency, but also has the best trade-off of high utility and strong robustness.

3 PROBLEM FORMULATION

3.1 Problem Definition

We try to provide a definition of the physical machine placement problem in the context of edge computing. The notations and their corresponding meanings in this paper are summarized in Table 3. Our objective is to determine the optimal placement of p physical machines, given N base stations and m mobile users within a specific area. The primary goal is to achieve a balance between utility and network robustness, which is also referred to as compromise programming or compromise planning in this paper. In the PESPU problem, it is assumed that all users fall within the coverage range of the base station, and all placement schemes inherently cater to the users' requirements.

Only if a user is within the coverage of one base station b_i , this user can access b_i :

$$u_j \in \text{cov}(b_i), \forall u_j \in U, \forall b_i \in B \quad (1)$$

If u_m is covered by b_i and b_j , then b_i and b_j are defined as cross coverage, and the number of users of cross coverage is defined as $\delta(b_i, b_j)$.

The robustness is the ability to provide the users with required service when runtime failures occur. Assuming that the probability of a single physical machine failure is f , then the network robustness between base station i and base station j is measured by the probability of physical

TABLE 3: Key Notations

Notation	Description
b_i	i th base station
p_i	i th physical machine
f_{CPU}	multi-tenancy benefits
m	finite number of mobile users
N	finite number of base stations
p	finite number of physical machines
s_i	computation task size of base station b_i
k_i	number of physical machines of base station b_i
f	the probability of a single physical machine failure
y_i	capacity of physical machine p_i
Y_i	sum of the physical machine capacity on b_i
UT	average utility
R	network robustness
B	finite set of base stations
u_i	mobile user u_i
U	set of mobile users with size m
$\delta(b_i, b_j)$	number of users covered by b_i and b_j

machine failure and the user numbers in the cross coverage area:

$$r_{ij} = \delta(b_i, b_j)(1 - f^{k_i + k_j}) \quad (2)$$

where $i \neq j$, k_i is the number of physical machines of base station b_i and k_j is the number of physical machines of base station b_j . $\delta(b_i, b_j)$ represents the number of users cross-covered between base stations i and j . As $\delta(b_i, b_i)$ is meaningless, we set its value to 0. The overall robustness is:

$$R = \frac{\sum_{a=1}^N \sum_{b=1}^N [f(\sum_{i=1, j=a}^p x_{ij} + \sum_{i=1, j=b}^p x_{ij})\delta(b_i, b_j)]}{2} \quad (3)$$

Since physical machines are placed together, there will be a performance loss, so a non-centralized placement scheme can get better performance. Here we use multi-tenancy benefits to define performance. According to [13], multi-tenancy benefit is expressed by the formula of CPU utilization:

$$f_{CPU_i} = -\log_{s_i} y_i \quad (4)$$

where s_i ($0.9 < s_i < 1.0$) is determined by the computation task size, y_i ($y_i > 1$) is the maximum number of users served by each physical machine. In general, performance increases as y_i increases, because multi-tenancy enables a higher degree of resource sharing. But when the number of users increases to a certain level, the performance slows down and converges. The capacity of a physical machine shows the max users that the physical machine can serve. Here, Y_i represents the total capacity of the physical machines on the base station b_i :

$$Y_i = \sum_{p_i \in B_i} y_i \quad (5)$$

Utility refers to the performance obtained by actual users near the base station from the edge server, such as CPU performance. According to Eq. (4) and Eq. (5), we give the formula for the average utility as follows:

$$UT = \sum_{i=1}^N \left(\frac{-\log_{s_i} Y_i}{\text{cov}(b_i)} \right) \quad (6)$$

Our objective is to find an optimal edge server placement solution in order to maximize the compromise planning of the utility and network robustness, thus providing a better user experience for mobile users. The calculation formula is:

$$O = \sqrt{\omega_{ut}(\frac{UT_z^* - UT_z}{UT_z^* - UT_{\bar{z}}})^2 + \omega_r(\frac{R_z^* - R_z}{R_z^* - R_{\bar{z}}})^2} \quad (7)$$

where UT_z^* and R_z^* are the *hypothetically-worst* PESPU strategy, i.e., UT_z^* is the strategy with minimum utility and R_z^* is the strategy with minimum robustness. We use ω_c and ω_r as weights to represent the priorities for utility and robustness. $UT_{\bar{z}}$ is the average utility of all strategies, and $R_{\bar{z}}$ is the average robustness of all strategies.

PESPU problem's objective can be expressed as:

$$\text{maximize } O \quad (8)$$

3.2 Problem Hardness

In this section, we try to prove that the PESPU problem is $\mathcal{NP}$ -hard. In general, to prove the hardness of one problem, we can try to simplify it from a known $\mathcal{NP}$ -hard problem. If it can be simplified from a known $\mathcal{NP}$ -hard problem, it is also $\mathcal{NP}$ -hard. Here we use the *k Heaviest Subgraph* (*kHS*) problem [14] to help us prove that the PESPU problem is $\mathcal{NP}$ -hard.

Given a graph $G = (V, E)$ with a weight function ω , the *kHS* problem is formulated below [12]:

$$\max_{G'=(V',E') \subset G} \sum_{(u,v) \in V'} \omega(u,v) \quad (9)$$

where $G' = (V', E')$ is a subgraph of G , $|V'|$ is the number of vertexes of G' ($|V'| = k$) and E' is the set of edges of G' .

Theorem 1. *The PESPU problem is a $\mathcal{NP}$ -hard problem.*

Proof. To prove that the PESPU problem is $\mathcal{NP}$ -hard, we try to reduce the $\mathcal{NP}$ -hard *kHS* problem to the PESPU problem. The process of simplification is as follows. For a *kHS* problem, we can obtain an instance $kHS(V, E, \omega, k)$. Now we structure an instance $PESPU(n, E^*, C, p)$. Given n edge servers and p physical machines, each physical machine $p_i, p_j \in V$. Let $V = n, K = p$ and the weight of the i th vertex in V is C . Eq. (6) calculates the utility of a physical machine as its weight. If the physical machines p_i and p_j are distributed on different base stations b_i and b_j , and b_i and b_j overlap, then there is an edge $e_{i,j}$ between physical machines p_i and p_j . In that case, $E^* = \{e_{i,j} | \delta(b_i, b_j) > 0\}$, and $E^* = E$. According to Eq. (7), we use P and R to calculate O . Therefore, the satisfaction of objective (9) and objective (8) is equivalent. The PESPU problem is reducible from *kHS* and it is thus $\mathcal{NP}$ -hard. $\square$

4 APPROACH

4.1 Integer Programming based Approach

First we should make the PESPU problem a Constraint Optimization Problem (COP). Its constraint is the number of physical machine and each machine can only be placed in one base station. And its optimization objective is the compromise planning of utility and robustness. Its decision variable is a finite set of variables $X = [x_{11}, x_{12}, \dots, x_{pN}]$, each variable's domain of definition is $D_{ij} = \{0, 1\}$, $i = 1, \dots, p, j = 1, \dots, N$. A solution to the problem should meet all constraints η . Given a constant τ of computation task size, capacity m , number of users covered by base station

b_i $cov(b_i)$, failure probability f of each physical machine, and a set of base station $B = \{b_1, \dots, b_N\}$, the utility can be computed by physical machine capacity:

$$UT = \sum_{j=1}^N \left(\frac{-\log_{\tau}(\sum_{i=1}^p x_{ij} m_i)}{cov(b_j)} \right) \quad (10)$$

where m is the physical capacity, representing the max users that can be accommodated.

The robustness is calculated from every two base stations in the area. So the robustness is determined by the number of physical machines, probability of physical machine failure and the number of users jointly covered. The robustness calculation formula of the whole area is as follows:

$$R = \frac{\sum_{j=1}^N f(\sum_{i=1}^p x_{ij}) \delta(i, j) f(\sum_{i=1}^p x_{ij})}{2} \quad (11)$$

where f is the probability of physical machine failure. Before calculating the compromise plan, we do some calculations to make the utility and robustness dimensionless. Then we take the sum of these two terms as the compromise planning. And the COP model for the PESPU problem can be expressed as follows [12]:

$$\text{maximize } \sqrt{\omega_{ut}(\frac{UT^* - UT}{UT^* - UT})^2 + \omega_r(\frac{R^* - R}{R^* - R})^2} \quad (12)$$

Constraints:

$$x_{ij} \in \{0, 1\} \quad (13a)$$

$$\sum_{j=1}^N x_{ij} = 1, \forall i \in \{i | 1 \leq i \leq p\} \quad (13b)$$

Constraint (13b) ensures that every physical machine is placed in one base station under constraint (13a).

The integer programming tool used in this paper is IBM CPLEX Optimizer³¹. In addition, we name the above optimal approach PESPU-O.

4.2 Heuristic-based Approach

In the context of a PESPU problem, the number of feasible solutions can be expressed as n^k , exhibiting exponential growth with an increase in the number of base stations (n) and edge servers (k). Consequently, the time required for the PESPU-O algorithm, proposed in the previous section, also escalates. However, as n and k reach a certain threshold, employing the aforementioned integer programming algorithm becomes impractical due to the inability to achieve desired outcomes within an acceptable timeframe. To determine the scale of the problem, we classify it as small-scale if the number of base stations is less than 10^2 , servers is less than 10^2 , and users is less than 10^3 . Otherwise, it is categorized as large-scale. When dealing with substantial datasets, we utilize a range of heuristic algorithms to obtain the PESPU solution, optimize user experience, and conduct a comparative analysis of the effectiveness and efficiency of different algorithms in Section 5.

1. <https://www.ibm.com/products/ilog-cplex-optimization-studio>

4.2.1 PESPU-Greedy

When dealing with large-scale datasets, our initial approach is to employ the greedy algorithm to tackle the problems. The underlying principle of this algorithm is to make locally optimal choices at each step of problem-solving, without considering the global optimization. Given that the PESPU problem requires selecting the optimal solution incrementally, we begin by considering the server perspective. Starting with a system devoid of any servers, we iteratively place servers on the base stations. At each step, the selection of a base station aims to maximize the compromise programming of the entire system at that point. The pseudo code of PESPU-Greedy is presented in Algorithm 1.

Algorithm 1 greedy algorithm PESPU-Greedy.

Require: PESPU problem (B, S, U);
Ensure: solution x of PESPU problem;

```

1: function PESPU-Greedy(B,S,U) ;
2:   for  $i = 0; i < k; i++$  do
3:     for  $j = 0; j < n; j++$  do
4:        $x[i][j] = 0$ ;
5:     end for
6:   end for
7:    $besty = 0$ ;
8:    $x[0][random(n)] = 1$ ;
9:   for  $i = 1; i < k; i++$  do
10:    for  $j = 0; j < n; j++$  do
11:       $temp_x = x$ ;
12:       $temp_x[i][j] = 1$ ;
13:       $temp_y = get\_UR(temp_x)$ ;
14:      if  $temp_y > besty$  then
15:         $besty = temp_y$ ;
16:         $bestid = j$ ;
17:      end if
18:    end for
19:     $temp_x[i][j] = 0$ ;
20:     $x[i][bestid] = 1$ ;
21:  end for
22:  return  $x$ ;
23: end function

```

The algorithm commences by initializing all elements of the decision variable to 0 and setting the compromise planning value of the decision variable to 0 (Lines 1-7). To determine the placement of the first server, a random base station is selected (Line 8). Subsequently, for each subsequent server, the algorithm iterates through all base stations within the entire system (Lines 9-12). In each scenario, it calculates the compromise planning value (Line 13) and selects the base station placement that yields the highest objective function value (Lines 14-20). In PESPU-Greedy, the algorithm iterates nk times. During each iteration, it calculates the objective function value of the feasible solution at that point. The complexity of one cycle is $O(n^2k)$, resulting in a time complexity of PESPU-Greedy of $O(n^3k^2)$.

4.2.2 PESPU-GA

In addition to the greedy algorithm, we have also developed PESPU-GA (Algorithm 2) based on the concept of ge-

netic algorithms. Genetic algorithms simulate natural evolutionary processes to search for optimal solutions. It treats the problem's solution as biological individuals, reproduces and evolves these individuals, eliminates solutions with low fitness, and ultimately obtains the problem's solution. In PESPU-GA, we encode the placement strategy as the genes of individuals within the algorithm, and then calculate the fitness of these individuals. The complete pseudo code for PESPU-GA is provided in Algorithm 2.

Algorithm 2 genetic algorithm PESPU-GA.

Require: PESPU problem (B, S, U, inisolution);
Ensure: solution x of PESPU problem;

```

1: function PESPU-GA(B,S,U,inisolution)
2:    $mpopulation = generate\_population(inisolution)$ ;
3:   while  $evolution < 100$  do
4:     for  $i = 0; i < num; i++$  do
5:        $fun[i] = get\_UR(mpopulation[i])$ ;
6:     end for
7:      $GA\_Select(mpopulation)$ ;
8:      $GA\_Cross(mpopulation)$ ;
9:      $GA\_Variation(mpopulation)$ ;
10:     $evolution++$ ;
11:  end while
12:  for  $i = 0; i < num; i++$  do
13:     $fun[i] = get\_UR(mpopulation[i])$ ;
14:  end for
15:   $Select\_best(fun)$ ;
16:  for  $i = 0; i < k; i++$  do
17:    for  $j = 0; j < n; j++$  do
18:       $x[i][j] = mpopulation[best][i * n + j]$ 
19:    end for
20:  end for
21:  return  $x$ ;
22: end function

```

The PESPU-GA algorithm begins by using the solution obtained from the Greedy algorithm as the initial solution to form the initial population (Line 2). The compromise planning value of each feasible solution is then calculated as the fitness of the individual (Lines 4-6). Based on their fitness, individuals are selected using the roulette method, with a certain number of individuals being chosen while the rest are discarded (Line 7). These selected individuals undergo crossover, where their gene codes are randomly combined to generate new individuals and form a new population (Line 8). The crossover process involves randomly selecting two individuals as parents and a specific edge server serial number as the crossover point. The server allocation method of one parent before the crossover point and the allocation method of the other parent after the crossover point are retained to create a new individual. To prevent premature convergence, each individual should participate in crossover no more than a predetermined number of times. Additionally, each individual in the population has a chance to undergo mutation, which involves randomly selecting a server and placing it on another base station (Line 9). Once the cycle of evolution is complete, the fitness of each individual in the final population is calculated. The individual with the best fitness is selected (Lines 12-14), and its decoding yields the best sub-optimal solution (Lines 15-

21). In PESPU-GA, the algorithm consists of generating the initial population, computing fitness, selection, crossover, mutation, and decoding. The time complexity of PESPU-GA can be obtained by summing up the complexities of each part, resulting in a time complexity of $O(n^4k)$.

4.2.3 PESPU-SA

Algorithm 3, PESPU-SA, is designed to find sub-optimal solutions using the simulated annealing algorithm. Simulated annealing algorithm begins with a higher initial temperature and gradually reduces the temperature based on a cooling coefficient. It combines the characteristics of probability jumps to randomly explore the solution space in search of the global optimal solution for the objective function. Simulated annealing is a randomized optimization algorithm. Its primary structure consists of an outer loop responsible for cooling and an inner loop for generating new solutions.

Algorithm 3 simulated annealing algorithm PESPU-SA.

Require: PESPU problem (B, S, U, inisolution);

Ensure: solution x of PESPU problem;

```

1: function PESPU-SA(B,S,U,inisolution);
2:    $tempbestx = get\_UR(inisolution)$ ;
3:    $tempbestx = inisolution$ ;
4:    $bestx = tempbestx$ ;
5:    $temperature = 100$ ;
6:   while  $temperature > 1$  do
7:     for  $iter - 1$ ;  $iter < 100$ ;  $iter++$  do
8:        $newx = randomswap(x)$ ;
9:        $newy = get\_UR(newx)$ ;
10:      if  $newy > tempbestx$  then
11:         $tempbestx = newy$ ;
12:         $tempbestx = newx$ ;
13:      probability judgement;
14:      end if
15:    end for
16:    if  $tempbestx > bestx$  then
17:       $bestx = tempbestx$ ;
18:       $bestx = tempbestx$ ;
19:    end if
20:     $temperature = temperature * decrease\_rate$ ;
21:  end while
22:  return  $x$ ;
23: end function

```

In PESPU-SA (Algorithm 3), we also utilize the solution obtained from the Greedy algorithm as the initial solution (Lines 2-3). The algorithm starts by performing initialization and setting the initial solution as the optimal solution (Lines 4-5). It then sets up an external loop (Line 6) and an internal loop (Line 7). Once the cooling cycle begins, a new solution is generated by making random changes to the current solution (Lines 8-9). The objective function values of the new solution and the old solution are compared, and the new solution is selected as the temporary best solution based on a probability judgment (Lines 10-14). After the internal loop, the temporary best solution is compared with the old best solution, and the solution with the higher objective function value is retained (Lines 16-19). The temperature

is then reduced, and the loop is repeated (Line 20). This process continues until the algorithm terminates, and the best solution obtained is the solution that is retained at the end. In PESPU-SA, the number of iterations of the algorithm is fixed. In each iteration, the objective function value of the feasible solution needs to be calculated. Therefore, the time complexity of PESPU-SA is $O(n^2k)$.

5 EXPERIMENTS

In this section, we conduct experiments using a real-world dataset to evaluate the approaches proposed in Section 4. Specifically, on the small-scale dataset Set, we run four algorithms: PESPU-O, PESPU-SA, PESPU-Greedy, and PESPU-GA. We also include PESPU-Utility and PESPU-Robustness as control experiments. PESPU-Utility is an algorithm that focuses on optimizing utility by maximizing it without considering the system's robustness. PESPU-Robustness, on the other hand, is an algorithm that aims to maximize the system's robustness without considering utility. For the experiments conducted on the small-scale dataset, we compare the results of the PESPU-O algorithm with various heuristic algorithms. We analyze the gap between their results and provide an explanation for the observed differences. However, on the large-scale dataset, the PESPU-O algorithm is not applicable due to the problem's large scale. As a result, we only run three heuristic algorithms: PESPU-Greedy, PESPU-GA, and PESPU-SA to find sub-optimal solutions for the problem.

5.1 Experiment Setup

The dataset used in the experiment is a widely used EUA dataset², which contains the geographical locations of 1465 real-world base stations in Melbourne, Australia. We adjust the coverage of each base station according to the density of users around the base station. We follow the practice of Cui et al. [12] and the specific measures are as follows. According to the users' density in an area, we divide the areas into high-density area, medium-density area and low-density area. In high-density area, a base station's coverage radius is around 600 meters, while in medium-density area about 2500 meters and low-density area around 7500 meters. The experiments on the small-scale dataset Set#1 are conducted in the central business district of Melbourne, with a total of 125 base stations. The experiments on the large-scale dataset Set#2 are conducted in Melbourne metropolitan.

TABLE 4: Dataset settings

		n	k	m
S	Set#1.1	5,10,...,35	4	80
M				
A	Set#1.2	20	1,...,8	80
L				
L	Set#1.3	20	4	20,...,160
L	Set#2.1	100,200,...,800	80	4000
A				
R	Set#2.2	400	10,...,80	4000
G				
E	Set#2.3	400	80	1000,...,8000

2. <https://github.com/swinedge/eua-dataset>

In order to comprehensively analyze the advantages and disadvantages of each algorithm, we simulate the scenario change of the PESPU problem by changing three parameters. The changed parameters are divided into the number of server base stations that can be placed (n), the number of servers that can be used (k), and the number of mobile users in the region (m). Specific data changes are listed in Table 4. According to Eq. (7), this paper also needs to set the values of parameter ω_c and ω_r to calculate the value of compromise planning. Different values are used to reflect the different priorities of edge infrastructure providers for system utility and robustness in different scenarios. In this paper, the experiments are conducted in a general way, and the priorities of system utility and robustness should be set to the same. Therefore, both ω_c and ω_r are set to 1 in this paper. On each dataset, each algorithm is repeated 100 times, and we take the average value as the final result. The hardware and software used in the experiment are as follows: Intel(R) Core(TM) i7-7500U CPU @2.70GHz (2.90 GHz max), 8GB RAM, Windows10 operating system, Java 1.8.0_311 and Eclipse-2021-09.

5.2 Effectiveness Evaluation

The effectiveness of an algorithm is determined by the quality of the placement strategy it discovers. Generally, a higher compromise planning value indicates a better user experience. Therefore, we use the results obtained by these algorithms to evaluate their effectiveness. On the small-scale dataset, the performance results of each algorithm are presented in Fig. 3, Fig. 4, and Fig. 5. Based on these results, it can be observed that in all cases, the compromise planning value obtained by our integer programming algorithm, PESPU-O, is the best among all the algorithms. The PESPU-O algorithm exhaustively searches through all feasible solutions for each PESPU scenario and selects the solution that maximizes the objective function as the optimal solution. The result obtained by the PESPU-O algorithm represents the actual optimal solution to the problem, providing the best user experience under the given constraints. The results obtained by the three heuristic algorithms are feasible solutions, and their compromise planning values should not surpass that of the PESPU-O algorithm. Hence, the results of the PESPU-O algorithm meet the expected outcomes. This paper also utilizes the results of the PESPU-O algorithm as a benchmark to evaluate the results of the three heuristic algorithms.

Based on the data presented in Fig. 3 to Fig. 5, among the three heuristic algorithms, PESPU-SA produces results that are closest to the optimal solution, achieving approximately 94% of the optimal solution. PESPU-GA follows closely behind, reaching around 93% of the optimal solution. PESPU-Greedy performs the worst, with results at approximately 89% of the optimal solution. The simulated annealing algorithm (PESPU-SA) yields the best results, followed by the genetic algorithm (PESPU-GA), and finally the greedy algorithm (PESPU-Greedy). Both PESPU-SA and PESPU-GA achieve more than 90% of the optimal solution, demonstrating their superior performance in solving the PESPU problem on small-scale datasets. Furthermore, in terms of result stability, all three algorithms perform well.

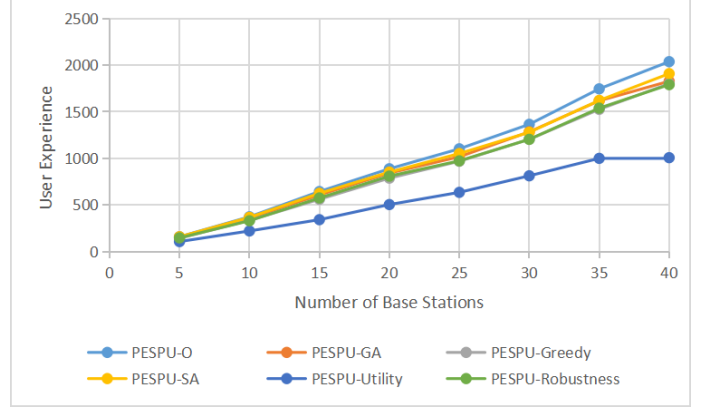


Fig. 3: User Experience vs. Base Station (Set#1.1)

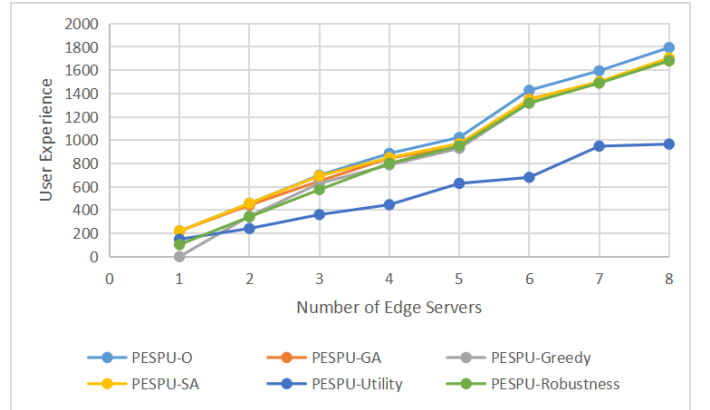


Fig. 4: User Experience vs. Edge Server (Set#1.2)

According to the data in Fig. 4, as the independent variable increases, the results of all three algorithms show minimal change. The difference between the results of the heuristic algorithms and PESPU-O remains relatively constant with an increase in the number of servers that can be placed. This indicates good stability in their performance. In general, the difference between the results of the three heuristic algorithms and the optimal algorithm does not fluctuate significantly with an increase in the number of independent variables. The difference remains stable within a fixed range.

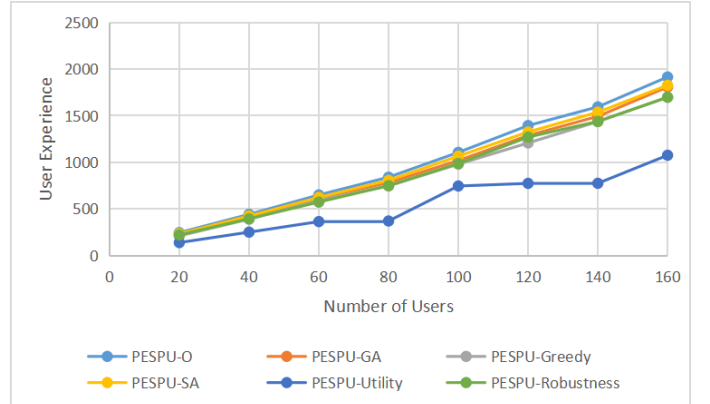


Fig. 5: User Experience vs. User (Set#1.3)

This provides confidence in the performance of the three heuristic algorithms when applied to large-scale datasets.

The differences in results between the three algorithms can be attributed to their respective solving approaches. PESPU-Greedy aims to find the optimal solution based on the current situation. However, this approach may only yield a local optimal solution, resulting in a compromise planning value that is lower than the other two heuristic algorithms. Therefore, it is expected that the results of PESPU-Greedy would be worse than those of PESPU-SA and PESPU-GA. Both PESPU-SA and PESPU-GA are meta-heuristic algorithms that search for the optimal solution on a global scale. They start with the solution obtained from PESPU-Greedy and further explore the search space to find a better solution. As a result, their results should be quite close to the optimal solution obtained by PESPU-O.

In conclusion, on the small-scale dataset, the PESPU-O algorithm can find the optimal solution, while among the three heuristic algorithms, PESPU-SA yields the best results, followed closely by PESPU-GA. These two algorithms achieve approximately 94% and 93% of the optimal solution, respectively, demonstrating their effectiveness in solving the PESPU problem. PESPU-Greedy, although yielding lower results compared to the other two algorithms, is still capable of solving the PESPU problem.

In the experiments on the large-scale dataset Set#2, the comparison of various algorithms in this paper is also similar to that in Set#1, except that the algorithm of PESPU-O can no longer be used, and its results can no longer be regarded as a benchmark. For Sets #2.1 to #2.3, the effectiveness of the three algorithms almost overlaps in whichever setting. The two curves representing the results of PESPU-Greedy and PESPU-SA overlap on the chart, and the other curve representing PESPU-GA is slightly different from them. Therefore, this paper selects the results on Set#2.1 for presentation and the results are shown in Fig. 6.

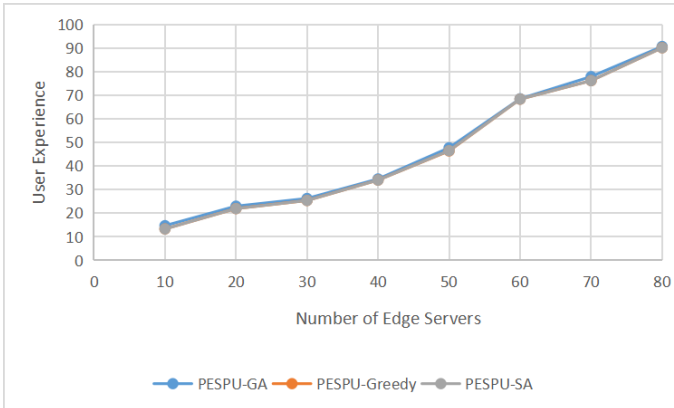


Fig. 6: User Experience vs. Edge Server on Set#2.2

According to the results of experiments on large-scale dataset, it can be seen that the experimental results of PESPU-Greedy and PESPU-SA are the same, and the results of PESPU-GA is slightly higher than them. The difference between them is about 1% in value. The change trends of the results of these three algorithms are consistent.

In the experiments on these six datasets, i.e., Sets #1.1 to #1.3 and Sets #2.1 to #2.3, with the increase of independent

variables, the compromise planning values obtained by all algorithms increase significantly. With the increase of the number of base stations, the distance between adjacent base stations becomes shorter, users in the area can be covered by more base stations, and the robustness of the system increases, thus improving the value of compromise planning. The increase in the number of servers in the same area can directly lead to the improvement of the overall utility of the system. Meanwhile, it also increases the number of servers available users, enhances the robustness of the system, and improves the value of compromise planning from two aspects. The growth of users number increases the number of users jointly covered by the base stations. According to Eq. (11), the robustness of the system increase, thereby improving the value of compromise planning.

5.3 Efficiency Analysis

In small-scale datasets, the time consumption of various algorithms is very low and similar. Therefore, in this section, we analyze the time consumption of three heuristic algorithms on large-scale datasets. The time consumption of each algorithm on three large-scale datasets are shown in Fig. 7, Fig. 8 and Fig. 9.

From Fig. 9, it can be observed that the number of users has minimal impact on the time consumption of the three algorithms. This is because the variation in the number of users only affects the values of elements in the robustness matrix of the algorithm and does not impact other calculation processes. Whether the number of users increases or decreases, it only alters the values of elements in Eq. (2) and Eq. (3), without affecting the complexity of the algorithm. On the other hand, as depicted in Fig. 7 and Fig. 8, the time cost of the algorithms increases significantly as the number of base stations or edge servers increases. In each algorithm, the strategy is represented by a two-dimensional array, with the size of the array determined by the number of base stations and servers. As the number of base stations and servers increases, the array capacity also increases. Consequently, it takes more time to calculate the utility and robustness, resulting in an increase in the overall time consumption of the algorithm.

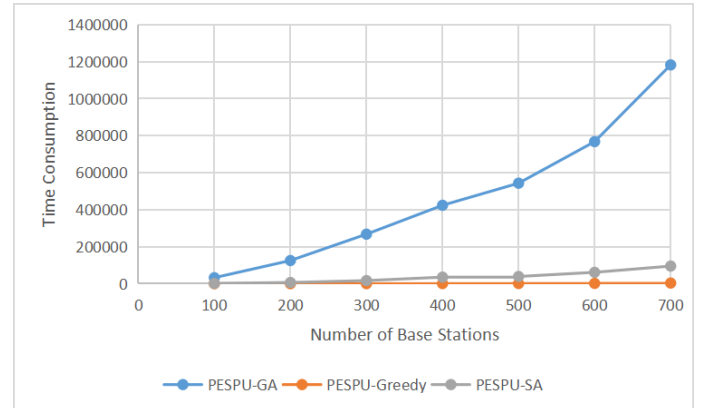


Fig. 7: Time Consumption vs. Base Station (Set#2.1)

After comparing the time consumption of the heuristic algorithms with the variation of independent variables,

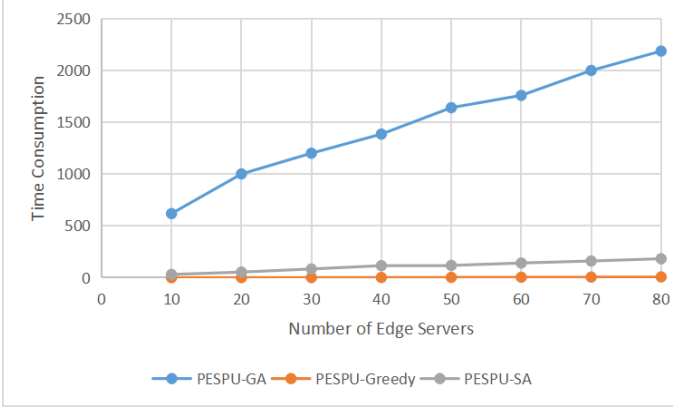


Fig. 8: Time Consumption vs. Edge Server (Set#2.2)

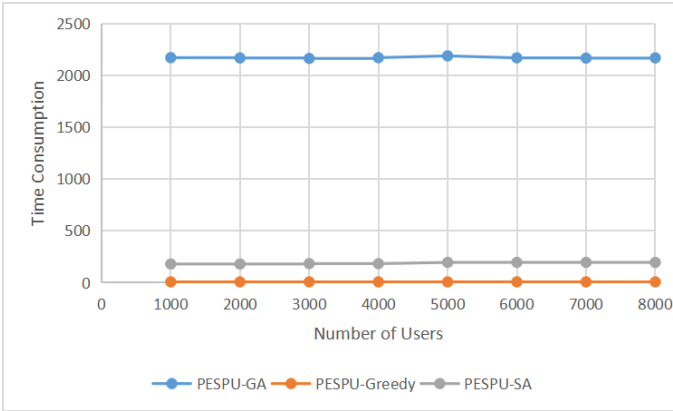


Fig. 9: Time Consumption vs. User (Set#2.3)

we can also compare the time consumption of the different heuristic algorithms themselves. Since PESPU-SA and PESPU-GA utilize the result of PESPU-Greedy as the initial solution, we need to add the time consumption of PESPU-Greedy to the time consumption of PESPU-SA and PESPU-GA. Based on Fig. 7, Fig. 8, and Fig. 9, the rankings of time consumption for the different heuristic algorithms remain consistent across the three datasets. PESPU-GA takes the longest time, followed by PESPU-SA, and then PESPU-Greedy. This aligns with their respective algorithm complexities. PESPU-SA and PESPU-GA's time consumption includes that of PESPU-Greedy, hence their overall time consumption is higher. The time complexity of PESPU-SA is $O(n^2k)$, as it only requires two inner and outer iterations, with just one linear operation performed in each cycle. The efficiency of PESPU-GA is often lower compared to other traditional optimization algorithms, which is an inherent disadvantage of genetic algorithms. The process of PESPU-GA is complex and involves extensive calculations for fitness evaluation, selection, and crossover operations. As a result, its time consumption is generally higher than other heuristic algorithms. Among all the algorithms, the PESPU-O algorithm has the highest time consumption, which increases exponentially with the independent variable. The PESPU-O algorithm needs to traverse all feasible solutions of the problem and calculate the utility and robustness for each feasible solution. In contrast, the three heuristic algo-

gorithms only need to calculate and compare a small portion of the feasible solutions during the calculation process. Therefore, their time consumption is significantly lower compared to that of the PESPU-O algorithm.

5.4 Discussion

In this section, the dataset was divided into small-scale datasets and large-scale datasets, and experiments were conducted respectively. On small-scale datasets, the four algorithms designed in this paper achieved acceptable results. Among them, the PESPU-O algorithm can get the optimal solution of the problem as expected, while the sub-optimal solution obtained by the heuristic algorithm PESPU-SA can reach about 94% of the optimal solution, followed by the PESPU-GA, which can reach 93% of the optimal solution. PESPU-Greedy can reach 89% of the optimal solution. All of PESPU-SA, PESPU-GA and PESPU-Greedy have excellent ability to obtain sub-optimal solutions on small-scale datasets, and have superior effectiveness for solving PESPU problems, which means that they can be used to solve PESPU problems. On large-scale datasets, these three algorithms can get a good feasible solution to the PESPU problem.

5.5 Threats to Validity

5.5.1 External Validity

The main threat to the external validity is the setting of experimental parameters. Parameters are the distribution of users in one area, the number of edge servers that need to be placed and the location and number of the base stations that can be used. It is no doubt that the scenarios simulated with the datasets can not represent all the possible PESPU scenarios in the real world. In order to minimize the threat, we change three parameters to simulate various PESPU scenarios. In this way, the results can be applied broadly based on existing results.

5.5.2 Internal Validity

The main threat to the internal validity of the evaluation is the comprehensiveness of our experiments. In the experiments of set#1 and set#2, we only changed one of the three independent variables at a time, while these three variables may change at the same time in the real world. Although these scenarios are not simulated in the experiment of this paper, their results can generally be predicted from existing results.

5.5.3 Conclusion Validity

The main threat to the conclusion validity is the lack of statistical tests [12]. In the experiment, we repeatedly run each scenario for 100 times and take their average values as the experimental results, which produces numerous test cases so as to reduce the practical significance of the experiment results [12]. Lack of statistical test may threaten the validity of the conclusion, but it is not significant.

6 RELATED WORK

Ericsson's Mobility Report [15] pointed out that the number of mobile devices would soar to 32 billion by 2023, and at that time people's daily life would be dominated greatly by these mobile and IoT devices. Limited by the physical size, computing capacities and energy capacities are these mobile devices' weaknesses. In 2012, Cisco³ promoted the edge server paradigm, which is also called fog computing. It can help mobile devices offload computation tasks to edge servers deployed at base stations or access points that are geographically close to them so that the limitation of mobile and IoT devices' computing performance and battery capacities can be tackled. Due to the emergence of edge cloud computing (ECC), many new studies were carried out, such as edge user distribution [16], [17], [18], edge application placement [19], edge server placement [12], [20] [21], etc.

Mobile users rely on edge servers located at base stations to receive service. Consequently, the selection of optimal edge server locations becomes a critical challenge. Existing research categorizes edge server placement problems based on various optimization objectives. These objectives encompass minimizing latency, reducing server deployment costs, optimizing user connections within clusters, maximizing coverage, and minimizing energy consumption, among others. Furthermore, specific parameter constraints need to be taken into account for different scenarios. These constraints include the number of edge servers, the capacity of individual servers, and the geographical placement of servers.

Numerous metrics are commonly employed to assess the effectiveness of edge server placement. These metrics encompass robustness, coverage, latency, performance, cost efficiency [22], throughput, load balancing, and energy consumption [11]. Several studies have focused on minimizing costs [23] [24] [25], optimizing edge server energy consumption [11], and reducing latency [26] [27]. Furthermore, there are research endeavors that aim to achieve multi-objective optimization, such as providing low-latency services while ensuring cost-effective deployments [28] [29].

Currently, there exist various algorithms for addressing edge server placement challenges. These include k-means, hierarchical tree structures, the least dominant set problem in graph theory, multi-objective constrained optimization, mixed-integer linear programming with heuristics, integer linear programming, k-means for mixed-integer quadratic programming, nonlinear mixed-integer programming, combining clustering with optimization, density-based clustering, and heuristics.

In recent years, the utilization of multi-objective joint optimization in edge server placement problems has gained significant traction. Li et al. [11] applied the concept of multi-objective optimization to analyze and formulate the ESP problem. They developed an energy-aware algorithm based on the particle swarm algorithm to seek the optimal solution for the ESP problem. Similarly, Liu et al. [27] devised a heuristic algorithm, employing a combination of the greedy algorithm and the Kuhn-Munkres algorithm, to tackle the multi-objective optimization of content cache and server placement in edge computing environments. This

approach effectively minimizes average delay and average bandwidth resource usage. On the other hand, Zeng et al. [30] focused on minimizing placement costs by utilizing linear programming to model the kESP problem, subsequently solving it through heuristic algorithms. Additionally, He et al. [12] proposed ESP-A, an approach specifically designed to address the ESP problem within large datasets. This approach evaluates the quality of the placement strategy based on network robustness and user coverage.

Previous research studies [12] [20] [25] [11] often simplified the ESP problem by disregarding the performance differences and capacity constraints of edge servers at each base station. However, this simplification does not accurately reflect real-life scenarios since a single server cannot cater to an unlimited number of users. There are studies that focus on server performance, such as Nahum et al. [31], who examined the impact of wide area network (WAN) conditions on WWW server performance. Additionally, Tao et al. [32] explored energy efficiency with performance guarantees in edge cloud computing (ECC). They addressed the need for low-energy and high-performance tasks by formulating an energy minimization optimization problem for mobile edge cloud computing.

In the context of edge cloud computing (ECC), the capacity of an edge server is constrained, and its performance directly affects the quality of service delivered to mobile users. This paper introduces the concept of average utility and network robustness and presents PESPU-O, a solution to tackle the PESPU problem. Additionally, three heuristic algorithms are proposed to effectively address the PESPU problem and formulate suitable server placement strategies. These approaches successfully resolve the PESPU problem and enable the development of optimal server placement strategies.

7 CONCLUSION AND FUTURE WORK

Based on the existing research on the placement of edge servers, this paper brings the average utility and network robustness into the research scope for the first time. From the perspective of edge server, each server is composed of a certain number of physical machines. Different combinations of physical machines will lead to different user experience. Considering average utility and network robustness of the whole system, the problem for finding out the most appropriate server combination placement strategy, named as the PESPU problem. In this paper, we model the server combination placement problem for average utility and network robustness for the first time, and prove that this problem is NP-hard. To solve the PESPU problem, we put forward an optimization algorithm PESPU-O based on integer programming to find the best solution for the small-scale PESPU problem. For the large-scale PESPU problem, we design three heuristic algorithms to find the solution of the problem. In this paper, a well known dataset is used to evaluate the effectiveness and efficiency of these algorithms. From the experimental results, the PESPU-SA and PESPU-GA algorithms designed in this paper have superior results on solving the PESPU problem on small-scale dataset, the designed three algorithms can obtain a high approximation to the optimal solution. And on large-scale dataset, the

3. <https://www.ericsson.com/en/reports-and-papers/mobility-report>

three algorithms can obtain a good solution for the PESPU problem. This paper gives a feasible solution to the PESPU problem.

As the first attempt to solve the PESP problem considering both average utility and robustness, we focus on trade-off placement with the aim of better optimizing the user experience. This will lay a solid foundation for further research on edge server placement issues focusing on user experience, and open up some new directions for combinatorial optimization. In the future, we will study the problem of optimizing the deployment of edge servers in specific regions considering various other real-world factors.

ACKNOWLEDGMENTS

This paper is particularly grateful to Professor Qiang He and Dr. Guangming Cui from Swinburne University of Technology for their help. We also sincerely thank the anonymous reviewers for this paper.

AUTHORS' INFORMATION

Shuliang Wang received his Ph.D. in spatial data mining from Wuhan University and Hong Kong Polytechnic University, in 2002. He is currently a Professor of Beijing Institute of Technology, Executive Dean of the E-Government Institute. He acts as the Vice Chairman of the Chinese Manufacturing Enterprise Double Innovation Development Alliance, Chief Scientist of the National Key Research and Development Program. His research interests include spatial data mining, and big data intelligence. He was awarded the first prize of National Science and Technology Award for Science and Technology Progress, National Excellent Doctoral Dissertation, CICC Young Scientist Award and so on.

Shuo Liu received her bachelor's degree from Shandong University, China, in 2020. She is a master's student at the Beijing Institute of Technology. Her research interests include big data intelligence, edge computing, and deep reinforcement learning.

Yijie Yang studies at school of computer science and technology, Beijing Institute of Technology. His research interests include software engineering and edge computing.

Hanning Yuan received her Ph.D. on data mining from Wuhan University. She is currently an Associate Professor of Beijing Institute of Technology. Her research interests include data mining and big data. She was awarded the first prize of Hubei Science and Technology Award for Science and Technology Progress and so on.

Sijie Ruan is currently an assistant professor at the School of Computer Science and Technology, Beijing Institute of Technology. He received his Ph.D. degree and B.E. degree from Xidian University in 2022 and 2017, respectively. His research interests include spatio-temporal data mining and management, volunteered geographic information, and urban computing.

Yun Yang received his Ph.D. from the University of Queensland, Australia, in 1992, in computer science. He is currently a full professor in the Department of Computing Technologies at Swinburne University of Technology,

Melbourne, Australia. His research interests include software technologies, cloud and edge computing, workflow systems, and service computing.

DECLARATIONS

- **Funding**
The work is funded by the National Key R&D Program of China (2020YFC0832600), the National Natural Science Fund of China (62076027), and Australian Research Council Discovery Project (DP180100212).
- **Conflict of interest/Competing interests**
The authors declare that they have no competing interests.
- **Ethics approval**
Not applicable
- **Consent to participate**
Not applicable
- **Consent for publication**
Not applicable
- **Availability of data and materials**
Not applicable
- **Code availability**
Not applicable
- **Authors' contributions**
Shuo Liu and Yijie Yang designed the improved framework and conducted experiments under the supervision of Shuliang Wang, Sijie Ruan and Yun Yang. All authors participated in the data analysis and manuscript writing. The authors read and approved the final manuscript.

APPENDIX

Abbreviations

MEC: Multi-access edge computing

PESP: Physical edge server placement

PESPU: User-oriented physical edge server placement

NP: Non-deterministic Polynomial

PESPU-O: Optimization method for user-oriented physical edge server placement

REFERENCES

- [1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [2] C. Ma and Y. Yang, "A battery-aware scheme for routing in wireless ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 60, no. 8, pp. 3919–3932, 2011.
- [3] T. Soyata, R. Muralidharan, C. Funai, M. Kwon, and W. B. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *2012 IEEE Symposium on Computers and Communications, ISCC 2012, Cappadocia, Turkey, July 1–4, 2012*. IEEE Computer Society, 2012, pp. 59–66.
- [4] P. C. Sheu, S. Wang, Q. Wang, K. Hao, and R. A. Paul, "Semantic computing, cloud computing, and semantic search engine," in *Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), 14–16 September 2009, Berkeley, CA, USA*. IEEE Computer Society, 2009, pp. 654–657.
- [5] M. Aloqaily, B. Kantarci, and H. T. Mouftah, "Multi-agent/multiobjective interaction game system for service provisioning in vehicular cloud," *IEEE Access*, vol. 4, pp. 3153–3168, 2016.
- [6] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wirel. Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

- [7] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
- [8] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks *et al.*, "6g white paper on edge intelligence," *arXiv preprint arXiv:2004.14850*, 2020.
- [9] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 55–67, 2022.
- [10] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distributed Comput.*, vol. 127, pp. 160–168, 2019.
- [11] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *2018 IEEE International Conference on Edge Computing, EDGE 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE Computer Society, 2018, pp. 66–73.
- [12] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between user coverage and network robustness for edge server placement," *IEEE Transactions on Cloud Computing*, 2020, doi:<https://doi.org/10.1109/tcc.2020.3008440>.
- [13] G. Velkoski, M. Simjanoska, S. Ristov, and M. Gusev, "Cpu utilization in a multitenant cloud," in *Eurocon 2013*. IEEE, 2013, pp. 242–249.
- [14] M. Letsios, O. D. Balalau, M. Danisch, E. Orsini, and M. Sozio, "Finding heaviest k-subgraphs and events in social media," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 113–120.
- [15] N. Heuvelod, "Ericsson mobility report," *Ericsson, Stockholm*, 2017.
- [16] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distributed Syst.*, vol. 31, no. 3, pp. 515–529, 2020.
- [17] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. G. Hosking, J. C. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Service-Oriented Computing - 17th International Conference, ICSOC 2019, Toulouse, France, October 28-31, 2019, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11895. Springer, 2019, pp. 86–101.
- [18] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. G. Hosking, J. C. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Service-Oriented Computing - 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11236. Springer, 2018, pp. 230–245.
- [19] F. Chen, J. Zhou, X. Xia, H. Jin, and Q. He, "Optimal application deployment in mobile edge computing environment," in *13th IEEE International Conference on Cloud Computing, CLOUD 2020, Virtual Event, 18-24 October 2020*. IEEE, 2020, pp. 184–192.
- [20] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented k edge server placement," in *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020, Melbourne, Australia, May 11-14, 2020*. IEEE, 2020, pp. 81–90.
- [21] D. Lu, Y. Qu, F. Wu, H. Dai, C. Dong, and G. Chen, "Robust server placement for edge computing," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, May 18-22, 2020*. IEEE, 2020, pp. 285–294.
- [22] L. Chang, X. Deng, J. Pan, and Y. Zhang, "Edge server placement for vehicular ad hoc networks in metropolitans," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1575–1590, 2022.
- [23] J. Meng, C. Zeng, H. Tan, Z. Li, B. Li, and X. Li, "Joint heterogeneous server placement and application configuration in edge computing," in *25th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2019, Tianjin, China, December 4-6, 2019*. IEEE, 2019, pp. 488–497.
- [24] N. Mohan, A. Zavodovski, P. Zhou, and J. Kangasharju, "Anveshak: Placing edge servers in the wild," in *Proceedings of the 2018 Workshop on Mobile Edge Communications, MECOMM@SIGCOMM 2018, Budapest, Hungary, August 20, 2018*. ACM, 2018, pp. 7–12.
- [25] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distributed Syst.*, vol. 28, no. 4, pp. 1031–1045, 2017.
- [26] L. Lovén, T. Lähderanta, L. Ruha, T. Leppänen, E. Peltonen, J. Rieki, and M. J. Sillanpää, "Scaling up an edge server deployment," in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2020, Austin, TX, USA, March 23-27, 2020*. IEEE, 2020, pp. 1–7.
- [27] Z. Liu, J. Zhang, and J. Wu, "Joint optimization of server placement and content caching in mobile edge computing networks," in *Proceedings of the 8th International Conference on Networks, Communication and Computing, ICNCC 2019, Luoyang, China, December 13-15, 2019*. ACM, 2019, pp. 149–153.
- [28] L. Zhao and J. Liu, "Optimal placement of virtual machines for supporting multiple applications in mobile edge networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, 2018.
- [29] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," in *27th International Conference on Computer Communication and Networks, ICCCN 2018, Hangzhou, China, July 30 - August 2, 2018*. IEEE, 2018, pp. 1–6.
- [30] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, p. 32, 2019.
- [31] E. M. Nahum, M. Rosu, S. Seshan, and J. M. Almeida, "The effects of wide-area conditions on WWW server performance," in *Proceedings of the Joint International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS/Performance 2001, June 16-20, 2001, Cambridge, MA, USA*, M. K. Vernon, Ed. ACM, 2001, pp. 257–267.
- [32] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wirel. Commun. Lett.*, vol. 6, no. 6, pp. 774–777, 2017.