Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing

Phu Lai[1], Qiang He[1(✉)], Mohamed Abdelrazek[2], Feifei Chen[2], John Hosking[4], John Grundy[3], and Yun Yang[1]

[1] Swinburne University of Technology, Hawthorn, Australia
{tlai,qhe,yyang}@swin.edu.au
[2] Deakin University, Burwood, Australia
{mohamed.abdelrazek,feifei.chen}@deakin.edu.au
[3] Monash University, Clayton, Australia
john.grundy@monash.edu
[4] The University of Auckland, Auckland, New Zealand
j.hosking@auckland.ac.nz

Abstract. In mobile edge computing, edge servers are geographically distributed around base stations placed near end-users to provide highly accessible and efficient computing capacities and services. In the mobile edge computing environment, a service provider can deploy its service on hired edge servers to reduce end-to-end service delays experienced by its end-users allocated to those edge servers. An optimal deployment must maximize the number of allocated end-users and minimize the number of hired edge servers while ensuring the required quality of service for end-users. In this paper, we model the edge user allocation (EUA) problem as a bin packing problem, and introduce a novel, optimal approach to solving the EUA problem based on the Lexicographic Goal Programming technique. We have conducted three series of experiments to evaluate the proposed approach against two representative baseline approaches. Experimental results show that our approach significantly outperforms the other two approaches.

Keywords: Optimization · Resource management · Edge computing
Bin packing

1 Introduction

In recent years, the world has witnessed a surge in the number of cloud and mobile network connected end-devices, including mobile phones, wearables, sensors and a wide range of Internet of Things (IoT) devices. According to Ericsson's Mobility Report [4], it is predicted that there will be around 32 billion of such connected devices by 2023. This has produced a great challenge for online service providers in terms of guaranteeing a reliable and low-latency connection to end-users, which is one of the key quality-of-service (QoS) requirements [12].

C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 230–245, 2018.
https://doi.org/10.1007/978-3-030-03596-9_15

To tackle this issue, Cisco [1] has proposed the fog computing paradigm – also called *edge computing* – in which computation, storage, and networking resources are pushed closer to the edge of the network by deploying a number of intermediate *edge servers* with closer proximity to *end-devices*. This paradigm offers lower network latency and greater scalability than the conventional centralized cloud computing paradigm. This is particularly important for high volume streaming applications or critical systems such as autonomous traffic systems, health care, or cloud gaming, which require real-time decision making. In edge computing, online service providers hire existing edge servers to host their services to serve their end-users. Thin clients – such as wearables, sensors or smart phones – all that have limited storage and computing capability, benefit from this architecture by the capability to offload intensive computing tasks to the distributed edge servers near them [17]. In this way, the central cloud is not required to perform all the computing tasks single-handedly, which is highly resource demanding and generally incurs long network latency for end-users. Usually, an edge server covers a specific geographical area so that the users within its coverage can connect to it via LTE, 4G or Radio Network [5]. A number of edge servers would be deployed in a distributed fashion (usually near cellular base stations [5]) so that they can cover different geographical areas. The coverages of adjacent edge servers usually partially overlap to avoid blank areas not covered by any edge server. A user located in the overlapping area can connect to one of the edge servers covering them (*proximity constraint*) that has sufficient computing resource (*capacity constraint*) such as CPU, bandwidth, or memory.

Edge servers' capacity, current workloads, coverages, the number of users to allocate and their proximity to end-users can be obtained or calculated at any time. Based on this information, while fulfilling the above constraints, an optimization goal must be achieved from a service provider's perspective – to minimize the number of edge servers used – in order to attain an optimal solution to the allocation of the service provider's users due to the pay-as-you-go pricing model applied in edge computing [12,17], which might incur higher costs when the number of edge servers used increases. Additionally, due to the aforementioned constraints, there might be a number of users that cannot be assigned to any edge servers. Those users will be connected directly to a central cloud server. Therefore, another optimization objective is to maximize the number of users allocated to hired edge servers.

We refer to the above problem as an *edge user allocation* (EUA) problem then model it as a *variable sized vector bin packing* (VSVBP) problem, a non-geometric generalization of the classical BIN PACKING (BP) problem. The EUA problem is critical in edge computing, however, has not been properly investigated. Solutions to the task allocation problem in cloud computing have been investigated in [8,14]. However, the edge computing architecture is different from cloud computing, i.e., distributed vs. centralized. In addition, the various constraints and dynamic information discussed above significantly differentiate the edge computing environment from the traditional cloud computing environment with many unique characteristics. Thus, the approaches for task allocation in

cloud computing are not suitable for solving the EUA problem, hence the need for a new approach. In this paper, we make the following major contributions:

- we have modeled and formulated the EUA problem as a VSVBP problem;
- we have developed an optimal approach for solving the EUA problem using the Lexicographic Goal Programming technique; and
- we have evaluated our approach against two representative baseline approaches with extensive experiments to demonstrate its effectiveness.

The remainder of the paper is organized as follows. Section 2 motivates this research with an example. In Sect. 3, we give a background of the VSVBP problem. Section 4 discusses the proposed approach, which is evaluated in Sect. 5. Section 6 reviews the related work. Section 7 concludes this paper.

2 Motivating Example

A representative example of edge computing applications is large-scale mobile gaming [6] - the fastest growing gaming model [10]. The cloud gaming model has made online game platforms, such as Hatch[1] and Sony PlayStation Now[2], more accessible for thin-client mobile players since the resource-expensive game instance is running on a remote cloud server. Consider an increasingly popular virtual reality game G, which requires a great amount of computing power for graphic rendering. Employing the traditional centralized cloud model helps thin-clients offload the heavy computation tasks; however, this approach introduces a huge network delay due to the long distance between players and cloud servers. Therefore, pushing the processing power closer to players with edge computing is a promising solution to this problem. Figure 1 shows an example of edge computing architecture that can be implemented in this scenario.

Assume there are four edge servers in a specific area that can be used to host game G. Each edge server covers a particular geographical area. Users who are outside the coverage of an edge server will not be able to connect to it (*proximity constraint*). For example, user u_4 cannot be assigned to edge server s_1 or s_4 and has to be allocated to either server s_2 or s_3. Furthermore, we need to take into account various *capacity constraints* such as bandwidth, memory, processing power, etc. In Fig. 1, each edge server has a limited computing capacity denoted as a vector $\langle CPUcore, memory, VRAM, bandwidth \rangle$. The aggregate workload generated by users on a server must not exceed the remaining capacity of that server. There are seven users within the coverage of edge server s_2 with a total workload of $\langle 7, 7, 3.5, 28 \rangle$, exceeding the remaining capacity of server $s_2(\langle 7, 8, 4, 25 \rangle)$. Thus, the game provider cannot assign all of these users to a single server s_2. Since users $u_1, u_2, \ldots, u_5$ are also covered by other edge servers, it is possible to allocate them to other servers to share the workload with server s_2. One potential solution would be to allocate users u_1, u_2 to server s_1, users

[1] https://www.hatch.live/.
[2] https://www.playstation.com/en-gb/explore/playstation-now/.

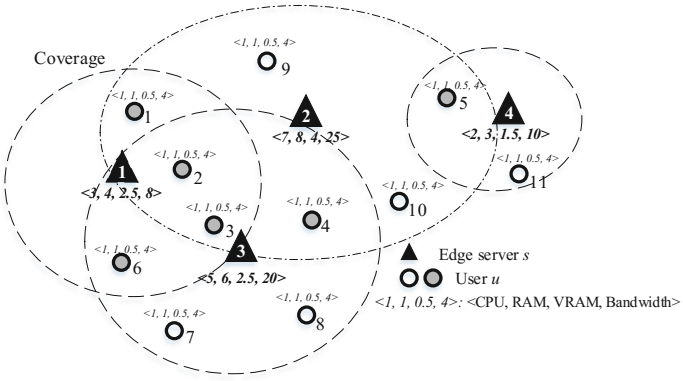


Fig. 1. Edge computing deployment example

u_3, u_6 to server s_3 and users u_4, u_5 stay with server s_2. No proximity or resource constraint is violated this way, but this might not be the optimal solution. If we assign users u_1, u_2, u_4 to server s_2, users u_3, u_6 to server s_3, and user u_5 to server s_4, server s_1 will no longer be required so the service provider can choose not to hire it to lower the total cost of hiring edge servers. This solution satisfies all the aforementioned constraints, uses the least servers to serve the most users, as well as guarantees the QoS.

3 Background

Definition 1 (Classical Bin Packing (BP) Problem). *Given an infinite supply of identical bins* $S = \{s_1, s_2, \ldots, s_i\}$ *with maximum capacity* $C = 1$ *and a set of* n *items* $U = \{u_1, u_2, \ldots, u_j\}$. *Let a value* $w_j \equiv w(u_j)$ *be the size of item* u_j *that satisfies* $0 < w_j \leq C$ *and* $1 \leq j \leq n$. *The objective is to pack all the given items into the fewest bins possible such that the total item size in each bin must not exceed the bin capacity* C: $\sum_{u_j \in U(s_i)} w_j \leq C, \forall s_i \in S$.

In the classical BP problem, one can normalize $C = 1$ without loss of generality since the bin capacity is just a scale factor. Aggregating item sizes not exceeding the capacity of the corresponding bin is the only constraint. This problem is known to be an $\mathcal{NP}$-hard combinatorial optimization problem [3].

Definition 2 (Variable Sized Bin Packing (VSBP) Problem. *Given a limited collection of bin sizes such that* $1 = size(s_1) > size(s_2) > \ldots > size(s_k)$, *there is an infinite supply of bins for each bin type* s_k. *Let* $L = \{s^1, s^2, \ldots, s^l\}$ *be the list of bins needed for packing all items. Given a list of items* $U = \{u_1, u_2, \ldots, u_j\}$ *with* $size(u_j) \in (0, 1]$, *the objective of the VSBP problem is to find an item-bin assignment so that the total size of the bins required* $\sum_{b=1}^{l} size(s^b)$ *is minimum.*

In the classical BP problem, all bins are homogeneous with a similar bin capacity. VSBP is a more general variant of the classical BP in which a limited collection of bin sizes is allowed. VSBP aims at minimizing the total size of the bins used, which is slightly different compared to the objective of the classical BP problem as discussed above.

Definition 3 (Vector Bin Packing (VBP) Problem). *Given a set of n items $U = \{u_1, u_2, \ldots, u_j\}$, the size of an item u_j is denoted as a d-dimensional vector $w_j = \langle w_j^1, w_j^2, \ldots, w_j^d \rangle$, $w_j \in [0, 1]^d$. One is given an infinite supply of identical bins $S = \{s_1, s_2, \ldots, s_i\}$ with maximum capacity $C = \langle 1^1, 1^2, \ldots, 1^d \rangle$. The objective is to pack the set U into a minimum number of bin s such that $\| \sum_{u_j \in U(s_i)} w_j \|_\infty \leq 1, \forall s_i \in S$.*

In the classical BP problem, the size of an item is presented as a single aggregation measure. By contrast, the size of an item in the VBP problem is associated with a multi-dimensional vector. The objective remains similar, in which the sum of packed item size vectors must not exceed the bin capacity vector in each dimension, which is normalized to 1 without loss of generality. The VBP problem is also known as multi-capacity BP problem in some literature.

In the EUA problem, a bin is referred to as an edge server with the maximum bin capacity being the remaining computing resource of that edge server. An item is referred to as an edge user, which can be a mobile phone or any IoT device; the size of an item is the amount of workloads generated by that user, measured by the computing resource needed to perform the requested task. In this paper, we tackle the EUA problem from a service provider's perspective. Thus, all users of an application generate the same amount of workload. In the real world, different edge servers may have different hardware specifications and host different applications for different numbers of users. Thus, they have different remaining server capacities, or computing resources. In addition, a computing task has various resource requirements such as CPU core, memory, video RAM, bandwidth, and so forth. Therefore, the amount of computing resource needed for a task should not be calculated by a just a single aggregate measure; instead, it can be represented as a d-dimensional vector where each dimension represents a resource type. The proposed VSVBP problem for EUA is $\mathcal{NP}$-hard since the classical BP, which is $\mathcal{NP}$-hard [3], is a special case of VSVBP where $d = 1$ and all the bins are identical in their capacity dimensionality.

4 Our Approach

4.1 Definitions

Edge servers have differentiated remaining capacity and multi-dimensional resource requirements for computation tasks. Therefore, the EUA problem can be modeled as a mixture of the VSBP problem and the VBP problem, hence a variable sized vector bin packing (VSVBP) problem. Our objective is to maximize the number of allocated users and minimize the number of hired edge servers.

We first introduce relevant notations and definitions used in our model in Table 1. In the EUA problem, every user covered by any edge server must be allocated to an edge server unless all the servers accessible to the user have reached their maximum capacities. If a user cannot be allocated to any edge servers, or is not positioned within the coverage of any edge servers, they will be directly connected to a service provider's central cloud server.

Table 1. Key notations

Notation	Description
$S = \{s_1, s_2, \ldots, s_i\}$	Finite set of edge server s_i, where $i = 1, 2, \ldots, m$
$C_i = \langle C_i^1, C_i^2, \ldots, C_i^d \rangle$	$d-$dimensional vector with each dimension C_i^k being a resource type, such as CPU utilization or disk I/O, representing the remaining capacity of an edge server s_i, $k \in \{1, 2, \ldots, d\}$
$U = \{u_1, u_2, \ldots, u_j\}$	Finite set of user u_j, where $j = 1, 2, \ldots, n$
$w_j = \langle w_j^1, w_j^2, \ldots, w_j^d \rangle$	$d-$dimensional vector representing the size of the workload incurred by user u_j. Each vector component w_j^k is a resource type, $k \in \{1, 2, \ldots, d\}$
$U(s_i)$	Set of users allocated to server s_i. $U(s_i) \subset U$
d_{ij}	Geographical distance between server s_i and user u_j
$cov(s_i)$	Coverage radius of server s_i

The total workload generated by all users allocated to an edge server must not exceed its remaining capacity (1). Otherwise, the server will be overloaded, causing service disruptions or performance degradation. Take Fig. 1 for instance. The aggregate workload incurred by users u_5 and u_{11} is $\langle 2, 2, 1, 8 \rangle$ does not exceed the remaining capacity of server s_4, $\langle 2, 3, 1.5, 10 \rangle$; therefore, this is a valid user-server assignment. If we allocate users $u_1, u_2, u_3, u_4, u_5, u_9, u_{10}$ to server s_2, it will be overloaded since the aggregate user workload $\langle 7, 7, 3.5, 28 \rangle$ exceed the server's remaining capacity $\langle 7, 8, 4, 25 \rangle$.

$$\sum_{u_j \in U(s_i)} w_j \leq C_i, \quad \forall s_i \in S \tag{1}$$

In the classical BP problem, an item can be placed in any bins as long as the bin has sufficient remaining capacity. However, in our problem, an edge server covers a limited surrounding geographical region. Thus, an item (user) can be assigned to specific bins (edge servers) since an edge server can only serve users who are located within its coverage (2). Take Fig. 1 for example. Server s_4 can serve users u_5 and u_{11} only. Since users might position in the overlapping areas of different edge servers, there is an optimal solution to allocate as many users as possible to as few servers as possible, which is the main focus of our research.

$$d_{ij} \leq cov(s_i), \forall i \in \{1, 2, \ldots, m\}; \forall j \in \{1, 2, \ldots, n\} \tag{2}$$

Our primary objective is to maximize the number of users allocated to all hired edge servers, which ensures the QoS from the service provider's perspective:

$$maximize \sum_{s_i \in S} |U(s_i)|, \tag{3}$$

Our secondary objective is to find a user-server assignment $\{u_1, \ldots, u_j\} \longrightarrow \{s_1, \ldots, s_i\}$ such that the number of servers hired E is minimum:

$$minimize\, E = |\{s_i \in S| \sum_{u_j \in U(s_i)} w_j > 0\}| \tag{4}$$

4.2 EUA Model

In this paper, we address the EUA problem with two optimization objectives: (1) maximizing the number of users allocated and (2) minimizing the number of edge servers hired, while satisfying the *capacity constraint* and *proximity constraint*. Accordingly, we model the EUA problem as a Lexicographic Goal Programming (LGP) problem [9]. In a lexicographic goal program, there are multiple optimization objectives with a number of constraints. These objectives are ranked by their levels of importance, or priorities. The solver will attempt to find an optimal solution that satisfies the primary objective and then proceed to find a solution for the next objective without deteriorating the previous objective(s). An LGP program can be solved as a series of connected integer linear programs. The LGP formulation of the EUA problem is as follows:

$$maximize \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} \tag{5}$$

$$minimize\ E = \sum_{i=1}^{n} y_i \tag{6}$$

subject to:

$$\sum_{j=1}^{n} w_j^k x_{ij} \leq C_i^k y_i, \forall i \in \{1, \ldots, n\}; \forall k \in \{1, \ldots, d\} \tag{7}$$

$$d_{ij} \leq cov(s_i), \forall i \in \{1, \ldots, n\}; \forall j \in \{1, \ldots, n\} \tag{8}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \forall j \in \{1, \ldots, n\} \tag{9}$$

$$y_i \in \{0, 1\}, \forall i \in \{1, \ldots, n\} \tag{10}$$

$$x_{ij} \in \{0, 1\}, \forall i \in \{1, \ldots, n\}; \forall j \in \{1, \ldots, n\} \tag{11}$$

where

$y_i = 1$ if server s_i is hired.
$x_{ij} = 1$ if user u_j is allocated to server s_i.
$cov(s_i)$ is provided by edge computing providers.

The objective (5) maximizes the number of users that are assigned to hired edge servers. The objective (6) minimizes the number of hired edge servers. Here, objective (5) is ranked higher than objective (6) in terms of priority. There are two groups of binary variables, i.e., x_{ij} (11) and y_i (10).

Capacity Constraint: As described by (7), each edge server s_j has a remaining capacity of $C_i = \langle C_i^1, C_i^2, \ldots, C_i^d, \rangle$, a d-dimensional vector. The aggregate workload of each resource type incurred by all allocated users must not exceed the corresponding remaining capacity of their assigned server. Take Fig. 1 for example. Assigning users u_5, u_{11} to server s_4 is valid since $\langle 2, 2, 1, 8 \rangle < \langle 2, 3, 1.5, 10 \rangle$.

Proximity Constraint: As described by (8), only users located within the coverage of an edge server can be allocated to the edge server. A user may be located in the overlapping coverage of multiple edge servers. For instance, users u_2, u_3 can be allocated to servers s_1, s_2 or s_3.

Constraint family (9) ensures every user is allocated to at most one edge server. In other words, a user can be allocated to either an edge server or service provider's cloud server.

5 Experimental Evaluation

In this section, we evaluate the performance of our approach by extensive experiments with a comparison to two baseline approaches. All the experiments were conducted on a Windows machine equipped with Intel Core i5-7400T processor (4 CPUs, 2.4 GHz) and 8 GB RAM. The LGP problem modeled in Sect. 4.2 was solved using IBM ILOG CPLEX Optimizer.

5.1 Baseline Approaches

Our approach will be benchmarked against two baseline approaches for user-to-server assignment, namely *random* and *greedy* approaches:

- *Random*: Each user will be allocated to a random edge server as long as that server has sufficient remaining capacity to accommodate the user and has the user within its coverage.
- *Greedy*: Each user will be allocated to an edge server that has the most remaining capacity and has the user within its coverage.

5.2 Experiment Settings

In this paper, we conduct experiments on data of base stations and end-users within the Melbourne central business district area in Australia, which has a total area of 6.2 km^2.

Experiment Data: We collect the location data of edge servers and end-users. Australian Communications and Media Authority (ACMA) publishes the radiocomms license dataset that contains the geographical location of all cellular base stations in Australia, which we will use as the locations of edge servers [5]. The coverage of each edge server is randomly set within a range of 450–750 m. In terms of end-users' locations, Asia Pacific Network Information Centre (APNIC) provides all IP address blocks allocated to Australia. We use an IP lookup service[3] to convert the obtained IP addresses into geographical locations. Since IP addresses in the last octet are likely to have identical geographical addresses returned by the IP lookup service, more end-users are uniformly generated around each of the obtained geographical locations. The raw experimental data has been made publicly available (EUA-dataset[4]).

Experimenting Parameters: In the experiments, we vary three setting parameters that may have an impact on our approach:

(1) **Number of end-users:** We randomly select different numbers of end-users $n = 4, 8, 16, \ldots, 512$. For each setting, we run the experiment 100 times to get 100 different random end-user distributions so that extreme cases, such as overly dense or sparse user distributions, are properly neutralized.
(2) **Number of edge servers:** The n end-users are located within the combined coverage of M edge servers. We assume that a total of m servers, where $m = 10\%, 20\%, \ldots, 100\% * M$, are available for accommodating those n end-users.
(3) **Remaining server capacity:** We experiment various levels of remaining server capacity based on the combined user workload. To be specific, we calculate 100%, 150%, ..., 300% of the combined user workload, then normally distribute it to M edge servers collectively covering the n end-users.

Performance Metrics: We evaluate the three approaches, namely our VSVBP, the random and the greedy baseline approaches, using the following metrics: (1) the percentage of allocated end-users of all end-users, the higher the better; (2) the percentage of hired edge servers of all available edge servers, the lower the better; and (3) the execution time (CPU time), the lower the better.

Given the data and the experiment parameters, we conduct three sets of experiments. The corresponding settings are described in Table 2. For each set, we vary one parameter and keep the other two fixed to observe the impact of each parameter on the approaches in the evaluation metrics.

In experiment set 1, the number of users vary from 4, 8, 16, 32, 64, 128, 256 to 512. All the edge servers, which have end-users within their coverage, can serve those end-users. The total remaining server capacity is 300% of the combined user workload. In experiment set 2, the number of users is fixed at 512, and the total remaining server capacity is fixed at 300% of the combined user workload.

[3] http://ip-api.com/.
[4] https://github.com/swinedge/eua-dataset.

Table 2. Experiment settings

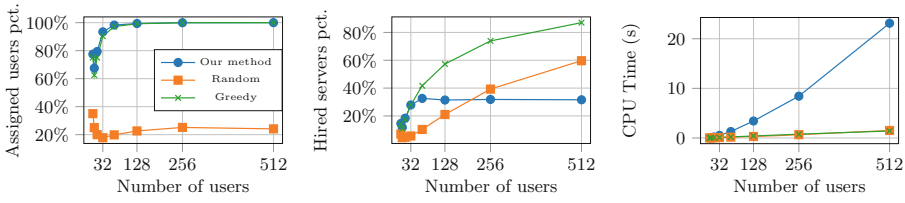
Factor	Number of users	Percentage of the total number of servers	Remaining server capacity
Set #1	$4, 8, \ldots, 512$	100%	300%
Set #2	512	$10\%, 20\%, \ldots, 100\%$	300%
Set #3	512	100%	$100\%, 150\%, \ldots, 300\%$

We change the number of edge servers that would be used to accommodate end-user, i.e., $10\%, 20\%, \ldots, 100\%$ of all edge servers to be made available for hire. In the last experiment set, we keep the number of users fixed at 512 and make all edge servers available for hire. The changing factor is the remaining server capacity – $100\%, 150\%, \ldots, 300\%$ of all users' workload combined.

5.3 Experimental Results and Discussion

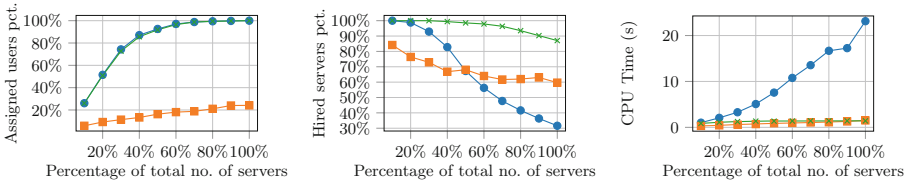
Figures 2, 3 and 4 show the results of the experiment set 1, 2 and 3, respectively. The three performance metrics are depicted in each sub-figure: (a) percentage of user allocated, (b) percentage of servers hired, and (c) execution time.

Figure 2 shows that in experiment set 1, as we increase the number of end-users from 4 users to 512, the random approach performs poorly in terms of allocated users percentage (only 20%–25% of the users are allocated) compared to the greedy approach and our approach, which give an equal performance with all users having been allocated. However, in terms of the number of edge servers hired, our approach starts to outperform the greedy approach as the number of end-users exceeds 32. The percentages of servers hired by the greedy and the random methods keep growing as the number of end-users increases, up to around 87.04% when serving 512 end-users. By contrast, our approach stably uses only around 32% of all available edge servers, 2.7 times less than that of the greedy approach, and remains steady even when the number of end-users increases from 32 to 512.



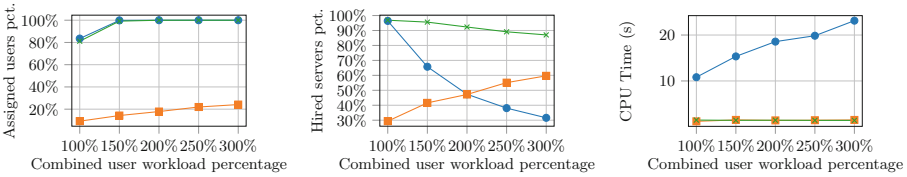
(a) Pct. of users allocated (b) Pct. of servers hired (c) Execution time

Fig. 2. Resulted metrics of set #1 (number of users changing)



(a) Pct. of users allocated (b) Pct. of servers hired (c) Execution time

Fig. 3. Resulted metrics of set #2 (number of servers changing)



(a) Pct. of users allocated (b) Pct. of servers hired (c) Execution time

Fig. 4. Resulted metrics of set #3 (remaining server capacity changing)

In experiment set 2, we change the number of edge servers available for hire. As depicted in Fig. 3(a), the allocated user percentage follows a similar trend as in experiment set 1. Regarding the percentage of hired edge servers (Fig. 3(b)), our approach continues to outperform the other two approaches as the number of edge servers increases.

Figure 4 shows that, in the last set of experiments, where the edge servers' total remaining capacity increases, we can observe the same trending patterns with our approach being the most effective out of the three approaches studied. As we increase the combined user workload percentage from 100% to 300%, our approach uses significantly fewer servers, dropping from 96.4% to 31.6% while the greedy method has to use around 90% of the all available servers.

Note that in all three set of experiments, the random approach seems to perform better than ours with fewer hired servers on some occasions. For example, when the number of users varies between 4 and 128 (Fig. 2(b)), when the total number of hired edge servers percentage varies between 10% and 50% (Fig. 3(b)), and when the total remaining server capacity changes between 100% and 200% (Fig. 4(b)). In fact, the random approach does not produce better results in these scenarios because although it uses fewer servers, the number of users allocated is extremely small (only 6%–20% of all end-users experimenting) compared to the other approaches, as shown in Figs. 2, 3, 4(a).

In terms of efficiency, the computation time of our approach increases considerably as we increase any one of the three parameters. In experiment set 1 with 512 users, the greedy and random methods take only approximately 1.5 s while our approach takes around 23.1 s to solve an instance of the EUA problem. This can also be observed in experiment sets 2 and 3, where we increase

the number of servers available for hired and the total remaining server capacity respectively. Since the EUA problem is an $\mathcal{NP}$-hard problem, it is expected that our approach, which optimally solves the problem, will take the most time as opposed to the other approaches, which can only make local decisions without considering the problem globally.

In general, increasing one of the three experimental parameters will increase the complexity of the EUA problem, which is an $\mathcal{NP}$-hard problem, and thus take more time to find an optimal solution. Our experimental results show that the random approach is not able to maximize the number of users allocated (the first optimization objective) as it can assign around only 20% of all the end-users in the experiments. The greedy approach is able to assign a similar number of end-users as our approach; the edge servers' adequate remaining capacities allow the greedy approach to find a capable edge server to accommodate most end-users. However, as shown in Figs. 2, 3, 4(b), our approach hires much fewer edge servers (the second optimization objective) than the greedy method to accommodate all the end-users. This is shown in all three experiment results, especially as the EUA problem scales up.

5.4 Threats to Validity

Threats to Construct Validity. The main threat to the construct validity in our study lies in the comparison with the two baseline methods, i.e., the random and greedy methods. The EUA problem studied in this research is a problem that has not been investigated before in this domain. Thus, we selected these two common and intuitive methods as baselines in our evaluation. Their designs are simple, especially the random method, which employs no heuristics. As a result, our approach is likely to obtain better experimental results, leading to a threat where the comparison with the selected baselines might not properly demonstrate the effectiveness of our approach in solving the EUA problem. To minimize this threat, we conducted experiments with three changing parameters as described in Table 2 to simulate different service deployment scenarios in the real world. This way, we could reliably evaluate our approach through both comparison with the baseline methods and also impacts of varying each experimental parameters on our approach.

Threats to External Validity. A major threat to external validity is whether our findings based on the experimental dataset can be generalized to other application domains in edge computing. Since there is currently no real-world dataset for this type of edge computing problems, we synthesized a dataset of edge servers and end-users based on reliable real-world data sources (ACMA and APNIC). However, this is a generic dataset, and it is possible that different application domains might have different factors that could impact the experimental results, such as the density and distributions of edge servers and end-users. Thus, our approach was evaluated across a breadth of problem scoping, varying in size, i.e., number of end-users, and complexity, i.e., number of edge servers and edge

servers' remaining capacities, to simulate as many types of edge server and end-user density and distribution as possible, as well as their combinations. This helped reduce the threat to the external validity of our evaluation and increased the generalizability of our results.

Threats to Internal Validity. A threat to internal validity of our work is the comprehensiveness of our experiments and whether or not the results are not biased by the experimental parameter settings. To mitigate this threat, we carried out extensive experiments with systematically selected parameters. The three experimental parameters (discussed in Sect. 5.2) are the three representative parameters that directly impact the outcomes of the approaches. Also, for each experiment set, we experimented with 100 different user distributions randomly selected from the pool of users to eliminate the potential bias caused by highly special scenarios such as overly dense or sparse distributions. Another threat to the internal validity of our evaluation is where more sophisticated scenarios could be simulated, e.g., those where two or more of those parameters change at the same time. In those scenarios, the results can be predicted in general based on the results that we have obtained. For example, if the total number of available edge servers and their total remaining capacities increase at the same time, the percentage of used servers of all hired by our approach will decline with a trend similar to but more significant than those shown in Figs. 3(b) and 4(b).

Threats to Conclusion Validity. The lack of statistical tests is the biggest threat to our conclusion validity. Statistical tests will be included in our future work to prove a statistically significant relationship between the experiment settings and the results. In this paper, we have compensated for this with meaningful comparison baselines and extensive experiments that cover many different scenarios, varying in both size and complexity. When an experimental parameter changes, the results are averaged over 100 runs of the experiment.

6 Related Work

Resource management in cloud computing has been extensively investigated in the last decade in many research tracks such as load balancing [7], virtual machine placement and provisioning [14], server and task allocation [8], etc.

Edge computing, or fog computing, is a new computing paradigm coined by Cisco in 2012 [1]. Edge computing is a natural extension of cloud computing with regard to the network topology and infrastructure deployment, where the architecture is more geographically distributed compared to cloud computing. This new architecture pushes the cloud resources closer to end-users. Barcelona, Spain is one of the first cities implementing edge computing with many applications, including power monitoring in public spaces, access control and telemetry of sensors, event-based video streaming, traffic analysis and regulation, and connectivity on demand [15]. There are more than 3,000 edge servers deployed across

the city serving thousands of IoT devices. The sheer number of edge servers and end-devices, with the horizontal scaling nature of edge computing, leads to the need for effective and efficient resource allocation solutions.

Chen et al. [2] proposed a distributed game theoretic computation offloading algorithm that was able to achieve a Nash equilibrium, minimizing the total energy consumption and offloading latency in the multi-channel mobile edge computing environment. By the proposed approach, they were able to optimally decide whether the users should offload computing tasks to an edge server and if yes, which wireless channel to be used for the computation offloading. In [16], Yao et al. tackled the problem of cost-effective edge server deployment using integer linear programming. They took into account the factors of resource capacity, user-server latency, and deployment costs. In their research, each edge server might not have the service installed to fulfill the requests from end-users. Thus, users' requests might have to travel across different edge servers until executed. However, they assumed that each server covers a region exclusively with other servers. They also assumed a predetermined edge server that first receives the user request. Our research targets more realistic edge computing scenarios where different edge servers' coverages might partially overlap. The authors of [13] also made an assumption that each small geographical area will only receive coverage from only a single edge server, which will be unlikely to happen in real-world scenarios. In [11], the authors formulated a problem similar to the EUA problem but with different objectives, which are to reduce task completion time and energy consumption. Yin et al. [18] addressed the edge server placement and provisioning problem with the objective of maximizing users coverage and minimizing network latency.

To the best of our knowledge, our work is the first to tackle the EUA problem in scenarios with multiple edge servers and end-users that possess and require multi-dimensional computing capacities. We also realistically and innovatively address this problem with respect to proximity constraints with the aims to maximize the number of allocated users and minimize the number of hired servers.

7 Conclusion

Edge computing is a promising new computing architecture, especially for high volume, data processing-intensive, latency-sensitive applications and services. However, when an edge computing scenario scales up, an ineffective edge user allocation solution will greatly increase the operational costs for service providers. To address this problem, we formulated the edge user allocation (EUA) problem as a variant of the bin packing problem named variable sized vector bin packing, an $\mathcal{NP}$-hard problem. We solved this problem using a Lexicographic Goal Programming technique with two optimization objectives, i.e., to maximize the number of users allocated and minimize the number of edge servers hired. We then conducted extensive experiments in scenarios with various service deployment requirements. Our experimental results show that our approach significantly outperforms two baseline approaches, greedy and random.

It is capable of allocating the most end-users with significantly fewer edge servers – nearly three times less than the greedy method – as the EUA problem scales up.

This research has established a basic foundation for the EUA problem and opened up a number of research directions. In our future work, we will take into account the users' mobility as well as the dynamics of users' computation tasks. In addition, apart from the proximity and capacity constraints, there are several elements that also play an important role such as network latency, service availability, pricing, and security.

Acknowledgments. This research is funded by Australian Research Council Discovery Projects (DP170101932 and DP18010021).

References

1. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC 2012, pp. 13–16. ACM, New York (2012). https://doi.org/10.1145/2342509.2342513
2. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2016). https://doi.org/10.1109/TNET.2015.2487344
3. Garey, M.R., Johnson, D.S.: Computers and Intractability, vol. 29. W. H. Freeman and Company, New York (2002)
4. Heuveldop, N.: Ericsson mobility report. Technical report, Ericsson, November 2017. https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-november-2017.pdf
5. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing a key technology towards 5G. Technical report 11, European Telecommunications Standards Institute (2015). http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf
6. Lin, Y., Shen, H.: CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. IEEE Trans. Parallel Distrib. Syst. **28**(2), 431–445 (2017). https://doi.org/10.1109/TPDS.2016.2563428
7. Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Trans. Parallel Distrib. Syst. **12**(10), 1094–1104 (2001). https://doi.org/10.1109/71.963420
8. Ren, R., Tang, X., Li, Y., Cai, W.: Competitiveness of dynamic bin packing for online cloud server allocation. IEEE/ACM Trans. Netw. **25**(3), 1324–1331 (2017). https://doi.org/10.1109/TNET.2016.2630052
9. Romero, C.: Handbook of Critical Issues in Goal Programming. Elsevier, Amsterdam (2014)
10. Smith, J.: The mobile gaming report. Technical report, Business Insider Intelligence (2016). http://www.businessinsider.com/the-mobile-gaming-report-market-size-the-free-to-play-model-and-new-opportunities-to-market-and-monetize
11. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi-server mobile-edge computing networks. CoRR abs/1705.0 (2017). http://arxiv.org/abs/1705.00704

12. Varghese, B., Wang, N., Nikolopoulos, D.S., Buyya, R.: Feasibility of fog computing. CoRR abs/1701.0 (2017). http://arxiv.org/abs/1701.05451
13. Wang, L., Jiao, L., Li, J., Mühlhäuser, M.: Online resource allocation for arbitrary user mobility in distributed edge clouds. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1281–1290, June 2017. https://doi.org/10.1109/ICDCS.2017.30
14. Wolke, A., Tsend-Ayush, B., Pfeiffer, C., Bichler, M.: More than bin packing: dynamic resource allocation strategies in cloud data centers. Inf. Syst. **52**, 83–95 (2015). https://doi.org/10.1016/j.is.2015.03.003
15. Yannuzzi, M., et al.: A new era for cities with fog computing. IEEE Internet Comput. **21**(2), 54–67 (2017). https://doi.org/10.1109/MIC.2017.25
16. Yao, H., Bai, C., Xiong, M., Zeng, D., Fu, Z.: Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. Concurr. Comput. **29**(16), 1–9 (2017). https://doi.org/10.1002/cpe.3975
17. Yi, S., Li, C., Li, Q.: A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata 2015, pp. 37–42. ACM, New York (2015). https://doi.org/10.1145/2757384.2757397
18. Yin, H., et al.: Edge provisioning with flexible server placement. IEEE Trans. Parallel Distrib. Syst. **28**(4), 1031–1045 (2017). https://doi.org/10.1109/TPDS.2016.2604803