Exploring Delay Reduction on Edge Computing Architectures from a Heuristic Approach

Hilal Alawneh[1,2], Jose David Nuñez-Gonzalez[1,2(✉)], and Manuel Graña[2]

[1] Department of Applied Mathematics, University of the Basque Country UPV/EHU, 20600 Eibar, Spain
hilal.alawneh.81@gmail.com

[2] Grupo de Inteligencia Computacional, University of the Basque Country UPV/EHU, 20001 San Sebastian, Spain

Abstract. Edge computing is a new promising paradigm helps users to execute their tasks on edge network which is closer to them rather than cloud. It can reduce application response time especially for those are critical to time such as healthcare applications, real-time apps, game playing, or traffic systems. Edge User Allocation (EUA) problem is responsible for allocate user application into edge servers on the edge network as app vendors' needing. In this paper, we propose a heuristic called Nearest Edge Server with Highest Capacity (NESHC) to solve the EUA problem. We use a real-word dataset in our extensive experiments. The results show that NESHC can reduce elapsed CPU time and outperform baseline approach (Optimal) and two state-of-the-art approaches (ICSOC19 and TPDS20). The reduction of delay causes increasing of user allocated into edge servers and leveraging the overall utilization of edge network system.

1 Introduction

Recently, the sector of Information Technology (IT) has developed dramatically with about 50-billion Internet of Things (IoT) devices will be connected to the Internet in the coming years [14]. This development will lead to the production of complex and computation intensive IoT applications that generate a big data [2]. Some limitations such as power and computational capabilities (i.e., CPU and memory) have caused downside effects on the execution of such resource-demanding applications on the devices [18]. Cloud computing is a technology that can support this growth by allowing on-demand access to a massive pool of computation resources [5,17]. However, the resources of cloud computing are centralized and far distance from IoT devices where the data generated and caused different problems regarding to applications that required critical conditions for execution such as reaching a high level of quality of services, low response time, ability to real-time interactive, and ensure a secure platform. Therefore, using

P. García Bringas et al. (Eds.): HAIS 2023, LNAI 14001, pp. 121–132, 2023.
https://doi.org/10.1007/978-3-031-40725-3_11

only cloud platform to execute such applications leads to some issues like security problems, generating a high delay and latency, interrupting the seamless of real-time interaction and violation of the quality of services [15]. Edge and fog computing have been used to provide more secure platform, reduce delay and latency, enable interaction for real-time applications, and enhance the quality of services by providing a pool of computational and storage capabilities at the edge and fog level of networks where they are close to IoT devices [4,6].

Edge computing is a new paradigm extending the cloud computing can reduce the latency for end-to-end devices, provide real-time interaction for the latency-sensitive applications where the edge servers that are much closer to end-user than cloud servers [12]. The process of determining which edge servers will serve which users is called Edge User Allocation (EUA), which is a critical problem in edge computing paradigm [12]. In edge computing paradigm, the computing, networking, and storage resources are span closer to the edge of the network by the number of edge servers which are closer to the end-users [3]. Edge computing provides lower network latency than the conventional cloud computing paradigm. This paradigm also offers a capability by enabling computation and storage at the edge level of the network. This substantially reduces data transfer (and bandwidth consuming) between the end user and the cloud [16]. Edge computing is especially important for extensive streaming applications or critical systems that require real-time decision-making, such as autonomous traffic systems, healthcare, or online gaming.

The edge servers are limited resources which might not be able to serve all the users within its coverage. Thus, some of the users might also be allocated in other coverage of the edge servers, or be allocated to the cloud. In order to ensure the Quality of Services (QoS) from the standpoint of the app vendor, the optimization target in this case is to maximize the number of users allotted to edge servers. Also, it's important to reduce the number of edge servers needed to serve those users. By lowering the cost of the app vendor's hiring edge servers to serve their app users [10], and increasing the resource usage on edge servers [7], this will guarantee the cost effectiveness of the allocation. One of the main goals in the server consolidation problem in cloud computing is to reduce the number of required servers [1,9]. Edge user allocation (EUA) problem is the term used to describe the aforementioned issue [10,11,13], by allocating user's task to the required edge servers in efficient way. The EUA problem is NP-hardness which becomes hard to solve effectively in an expanding edge computing environment. In this paper, we propose a heuristic approach called Nearest Edge Server with Highest Capacity (NESHC) to solve the EUA problem and reduce the delay of application execution on the edge network. The main contributions of this paper include:

- Proposes NESHC heuristic to solve NP-hardness of the EUA problem.
- Solving the complexity of the EUA problem by finding the optimal solution for user's task allocation into edge servers.
- Reduces the delay of execution time-aware applications such as real-time application.

– On a real-world dataset, extensive evaluations are performed to show that the efficiency of the proposed approach. The outcomes demonstrate that our approach outperforms some of baseline and the state-of-the-art approaches.

The remainder of the paper is organized as follows. Section 2 presents our proposed heuristic approach. Section 3 shows the results of the work compared with baseline and state-of-the-art approaches. The conclusion and future work are presented in Sect. 4.

2 Proposed Method

Edge user allocation (EUA) is NP-hard problem aims to find the best edge server for users for allocation where a dense distribution of edge servers and limited computing resources make it extremely computationally expensive to solve optimally the edge user allocation (EUA) problem in a large-scale scenario [12]. In some cases, the optimal solution would be high expensive, for example, if there are only 512 users and 125 edge servers, finding an optimal solution takes up to 23 s. This is unacceptable for applications or services that require real-time or near-real-time decisions. In order to efficiently find an optimal solution for the EUA problem and reducing the delay of applications execution, we introduce a heuristic approach called Nearest Edge Server with Highest Capacity (NESHC). The main aim of our heuristics is to reduce the delay in finding an optimal solution for edge user allocation problems to allow real-time applications or time-critical applications to complete their tasks within a required period of time. In this paper, we re-use the definition of the EUA problem which proposed by the authors in [10]. They set three definitions to handle with the EUA problem:

– Definition 1. Classical Bin Packing (BP) Problem: All bins are homogeneous with a similar bin capacity and the size of an item is presented as a single aggregation measure.
– Definition 2. Variable Sized Bin Packing (VSBP) Problem: This is a more general variant of the classical BP problem, where a limited collection of bin sizes is allowed and the objective is to minimize the total size of bins used, which is slightly different from the objective of the classical BP problem.
– Definition 3. Vector Bin Packing (VBP) Problem: 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. It is called as multi-capacity BP problem in some works.

For user tasks allocation, edge servers have differential remaining capacity and multi-dimensional resource requirements. As a result, the EUA problem can be represented as a hybrid of the VSBP and VBP problems, yielding a variable sized vector bin packing (VSVBP) problem. Our goal is to reduce the delay for the user allocation problem by finding the best edge server to allocate the user tasks and outperform both baseline and state-of-the-art approaches.

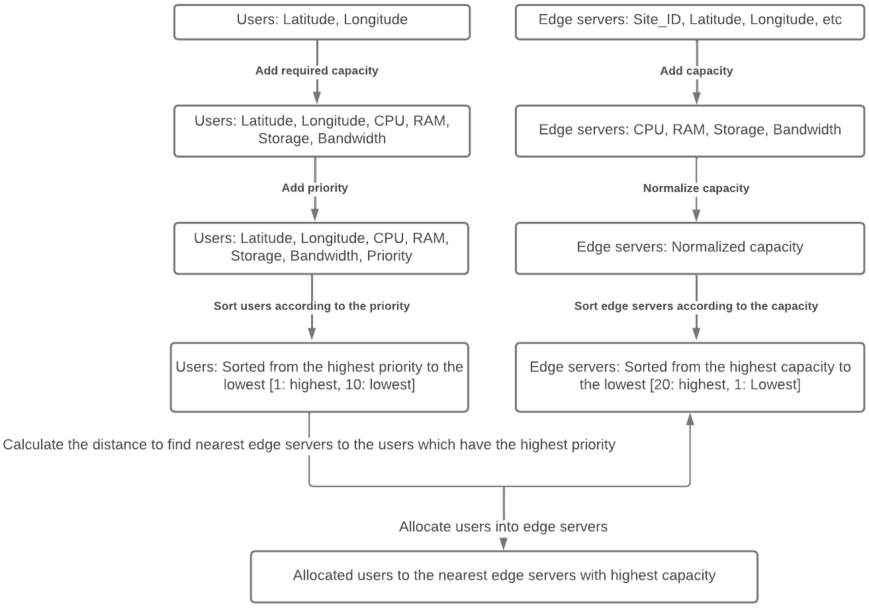

Fig. 1. The Flow Chart of the Heuristic Algorithm

As we can be seen in Fig. 1, our model considers the priority according to app vendor's needing and required capacity to serve user tasks and sorts users task in ascending manner from highest priority to lowest priority. After that, the capacity according to (CPU, RAM, Storage, and Bandwidth) for each edge servers will be randomly generated and sorted in descending order to find the highest capacity between all edge servers. The distance between the highest priority user task and the highest edge server capacity will be measured to find the nearest edge servers candidate to serve the user task in order to reduce overall delay and avoid sending user tasks to far distance edge servers or to the cloud.

Nearest Edge Server with Highest Capacity (NESHC)

In large-scale scenarios, finding an optimal solution will take very long time due to the problem's NP-hardness. In this section we propose a heuristic algorithm called Nearest Edge Server with Highest Capacity (NESHC) to solve the EUA problem. Suppose we have a set of users and a set of edge servers, we need to allocate user's task with highest priority to the nearest edge server with highest capacity. The edge servers will be sorted in descending order according to their capacity of CPU, RAM, Storage, and Bandwidth. Also, users will be sorted in ascending order according to their task's priority. The distance between users with highest priority and the edge servers with highest capacity will be calculated to find the nearest edge servers for users to be allocated. Given a set of edge servers S and a set of users U (lines 1–4), NESHC allocates an app vendor's users to edge servers. At the beginning all user tasks are unallocated. In line

5, the model sorts all edge servers according to their capacity of four vectors (CPU, RAM, Storage, and Bandwidth) in descending order to find the highest edge servers capacity where 1 is the lowest capacity and 20 is the highest capacity. The capacity will be normalized to be an integer number between 1 and 20. The model sorts all user tasks according to their priority generated by the app vendors in ascending order where 1 indicates to the highest priority and 10 indicates to the lowest priority (line 6). In line 7, the model calculates the distance between user u with highest priority and edge server s with highest capacity to find the nearest edge server s to the user u. In lines (8–15), NESHC allocates users which hold the highest priority to the nearest active edge server with the highest capacity, next to the nearest inactive edge server with the highest capacity. The process continue allocates all users from the highest priority to the lowest priority. Otherwise, if there are no free edge servers to serve user task, the user will be connected to the app vendor's cloud server to be allocated there line (16–18).

1. **initialization**
2. **a set of edge servers S and a set of users U**
3. **all users** u_j , $\forall u_j \in U$, **are unallocated**
4. **end initialization**
5. **sort edge server S in descending order according to their capacity of CPU, RAM, Storage, and Bandwidth, i.e. [20: highest, 1: lowest]**
6. **sort user u in ascending order of their priority, i.e. [1: highest, 10: lowest]**
7. **calculate the distance between user u with highest priority and edge server s with highest capacity to find the nearest edge server s to user u**
8. **for each user** $u_j \in U$ **do**
9. $Su_j \triangleq$ **user** u_j**'s nearest edge servers;**
10. $S_{u_j}^{active} \triangleq$ **user** u_j**'s active nearest edge servers;**
11. **if** $S_{u_j}^{active} != \emptyset$ **then**
12. **allocate user** u_j **to a nearest edge server** $s_i \in S_{u_j}^{active}$ **which has the highest capacity**
13. **else**
14. **allocate user** u_j **to a nearest edge server** $s_i \in S_{u_j}$ **which has the highest capacity**
15. **end if**
16. **if** s_i **cannot be decided then**
17. **allocate user** u_j **to the central cloud server**
18. **end if**
19. **end for**

3 Experiments and Results

All experiments are conducted on a Windows machine equipped with Intel Core i7-10510U processor (4 CPUs, 2.3 GHz) and 16 GB RAM. The software environ-

ment we utilized is Python 3.10.9 on windows 10 Pro. 64-bit. Python programming language is implemented using Python Jupyter Notebook[1], which can be freely downloaded from the Anaconda website[2]. The experiments are conducted on the publicly available EUA real-world dataset [3] [10], which contains the geographical locations of end-users and all cellular base stations in Australia.

We have conducted the experiments on a real-world dataset to evaluate the performance of our approaches against other baseline and state-of-the-art approaches. The five representative approaches, namely a Greedy baseline, a Random baseline, and three state-of-the-art approaches for solving the EUA problem:

- Greedy: This method allocates each user to the edge server with the largest available capacity, regardless of whether the server is active or not. Users are not allocated in any specific order.
- Random: This approach allocates each user to a random edge server available. Users are allocated in no specific order.
- ICSOC19 [11]: The proposed optimal approach will be used in our experimental evaluation. They proposed two approaches to tackled the EUA problem.
- TPDS20 [8]: The authors in this approach tackle the EUA problem by maximizing the number of allocated users while minimizing the overall system cost, which is determined using the expenses of required computing resources on edge servers. But TPDS20 does not provide dynamic QoS, users' QoS level are pre-specified at random.
- MCF [12]: Proposes an approach to solve the edge user allocation problem in large-scale scenarios. They aim to allocate the maximum number of users in minimum number of edge servers as possible.

We choose ICSOC19, TPDS20 and MCF as benchmark state-of-the-art approaches because they solved, the same problem which we handle in this paper, edge user allocation problem. Also, they used the same dataset EUA dataset that we use to conduct the experimental evaluation in this paper. In order to evaluate our proposed approach, we use the same experimental settings that handled in [12], which enable us to make a comparison between our proposed approach with aforementioned two baseline approaches and three state-of-the-art approaches. The simulation considers an urban area of $1.8km^2$ covered by 125 base stations, each station has one edge server. Each edge server covers a radius of 100–150m which is generated randomly. The results art taken by the average of repeated experiment 100 times to obtain 100 different user distributions. To evaluate the performance of our approach, we compare the CPU time taken to solve the EUA problem with results obtained by the six approaches (Optimal, Greedy, Random, ICSOC19, TPDS20 and MCF). The Optimal solution to a small-scale instance of the EUA problem can be found with an Integer Programming solver, e.g., Gurobi3 or IBM ILOG CPLEX4 [12]. The settings of the experiments are shown in Table 1.

[1] https://github.com/jupyter/.
[2] https://www.anaconda.com/.
[3] https://github.com/swinedge/eua-dataset.

Table 1. Experimental Settings [12]

	Users	Edge servers	Available resources (μ)
Set #1	100, ..., 1000	50%	35
Set #2	500	10%, ..., 100%	35
Set #3	500	50%	30, 35, ..., 75

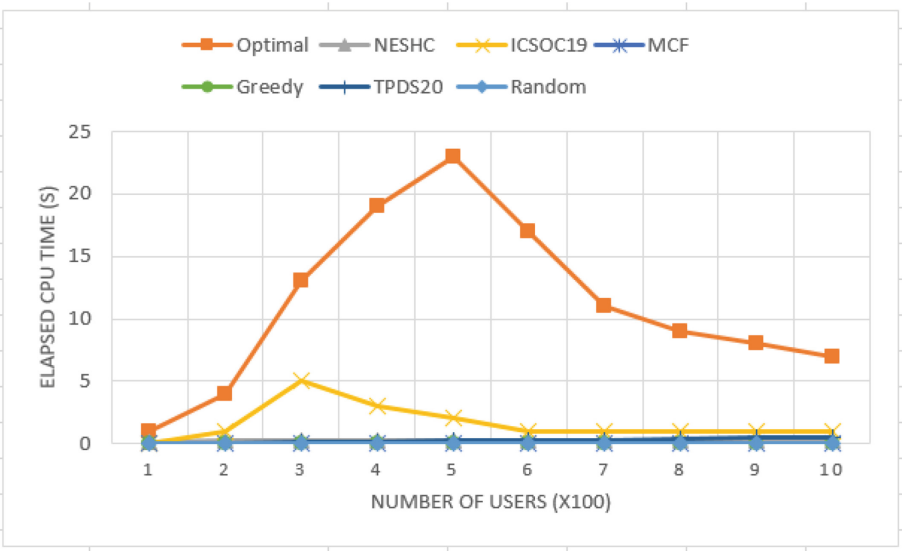

Fig. 2. Elapsed CPU time vs. Number of users (Set #1)

The results show that the NESHC approach can efficiently reduce overall delay for users tasks and enable time-aware applications to execute their requests within required time constraint. Figure 2 shows the efficiency of NESHC to reduce the average CPU execution time taken to solve an instance of the EUA problem. The maximum elapsed CPU time in seconds to solve the EUA problem is 0.308 s for NESHC at 1000 users, while the required time to solve the EUA problem by Optimal and ICSOC19 is up to 23 s and 5 s, respectively. NESHC can solve the EUA problem in 0.183 s when there are 100 users needed to be allocated. The authors of MCF indicated that MCF, Greedy, TPDS20, and Random required only 1–2 milliseconds to solve the EUA problem [12] and this is very short time and unreasoning to solve the EUA problem by allocating efficiently 1000 users to 125 edge servers. In Fig. 3, it can be seen that the Optimal takes up to 48 s to solve the EUA problem when the percentage of number of servers is 80%. ICSOC19 also takes 8 s at 60% of number of servers. The maximum elapsed CPU time for our approach NESHC is 0.506 s at 50% of number of servers. Figure 4 illustrates that both Optimal and ICSOC19 takes up to 30 s and 9 s at 45 and 55 available server capacity, respectively, to solve the EUA

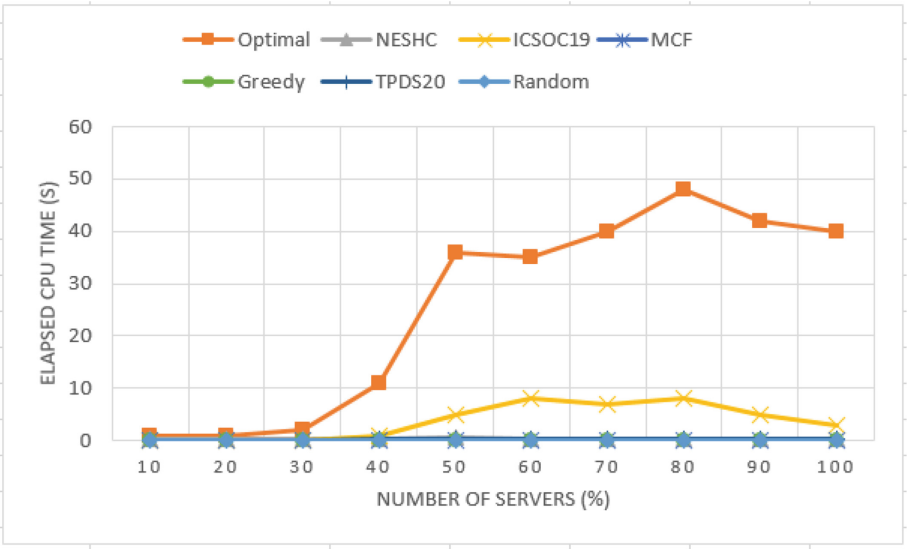

Fig. 3. Elapsed CPU time vs. Number of edge servers (Set #2)

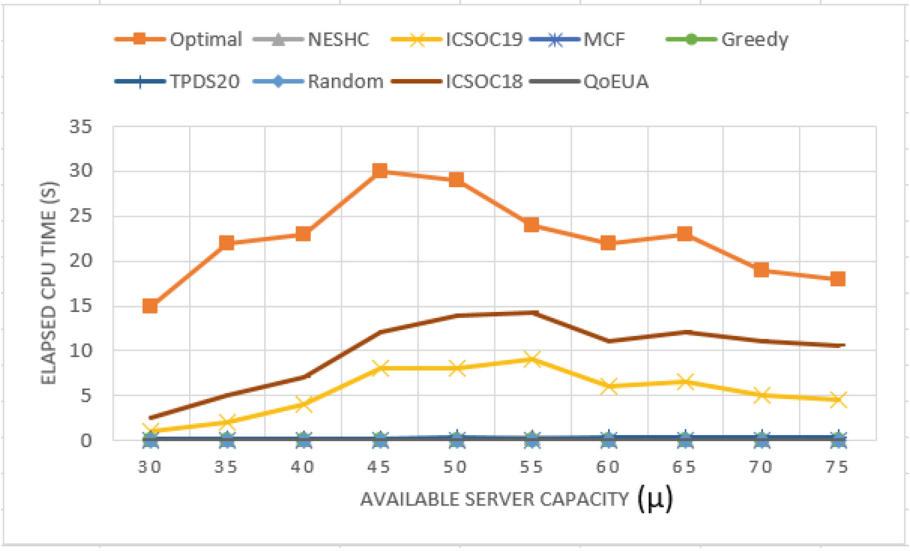

Fig. 4. Elapsed CPU time vs. Edge server capacity (Set #3)

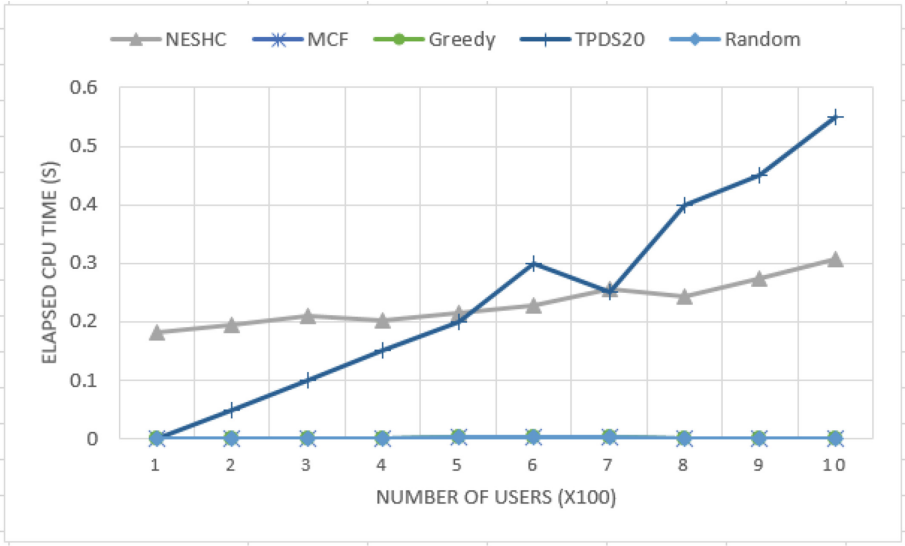

Fig. 5. Elapsed CPU time vs. Number of users (Set #1) without Optimal and ICSOC19

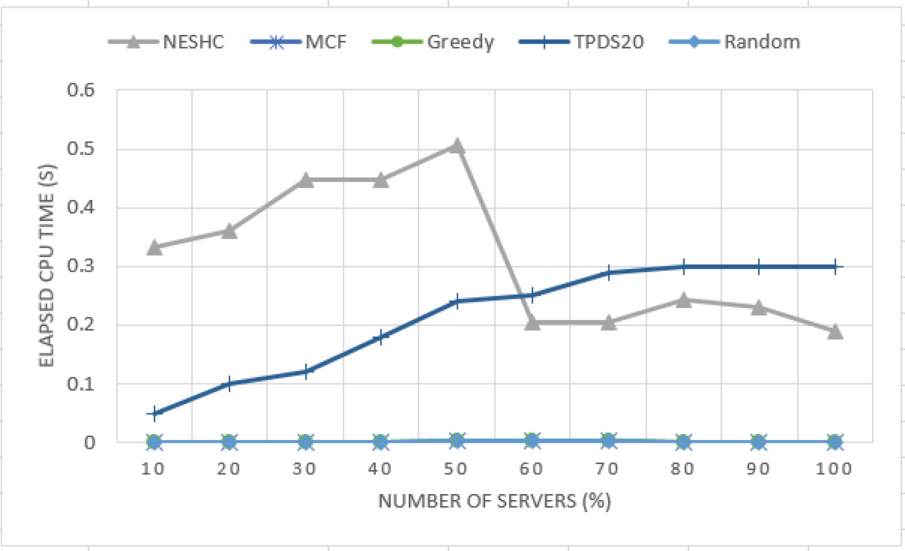

Fig. 6. Elapsed CPU time vs. Number of edge servers (Set #2) without Optimal and ICSOC19

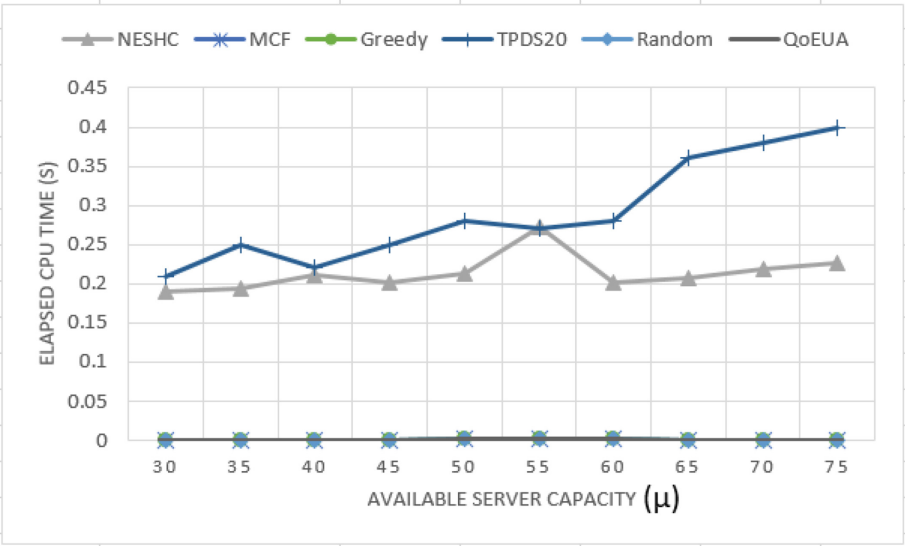

Fig. 7. Elapsed CPU time vs. Edge server capacity (Set #3) without Optimal and ICSOC19

problem. Figures 5–7 show the same results excluding Optimal and ICSOC19 in order to demonstrate the efficiency of NESHC compared with TPDS20. In Fig. 5, it can be clearly seen that TPDS20 takes up to 0.55 s at 1000 users, while NESHC needs only 0.308 s to solve the EUA problem at the same number of users. However, MCF, Greedy, and Random consume 1–2 milliseconds to tackle the EUA problem which is considered unreasoning time. In Fig. 6, NESHC takes 0.506 s of elapsed CPU time when number of servers is 50% while it falls down into 0.189 s when the percentage of servers is 100% which outperforms TPDS20. Figure 7 shows that NESHC outperforms TPDS20 along the period from 30 server capacity until 75 with 0.191 s to 0.227 s.

4 Conclusion

Edge computing can be used to solve the problem generated from sending user data into cloud for executing by reducing the delay and CPU execution time. The EUA problem is an NP-hard problem to achieve a mission of allocating efficiently users' tasks into some edge servers. Many user applications are critical to time such that healthcare applications which required a small amount of time to be served as well as real-time applications. Nearest Edge Server with Highest Capacity (NESHC) is a heuristic approach can efficiently solve the EUA problem and reducing the overall delay for user's tasks which be allocated into edge network. Delay reduction indirectly means that our system can increase the number of allocated users into edge servers in the edge network. Our experiments

on a real-world dataset show that NESHC outperforms the baseline approach (Optimal) and state-of-the-art approaches (ICSOC19 and TPDS20) in elapsed CPU time required to allocate users into edge servers. In the future, the system could be modified to utilize Fog network which provides more extensive resources capabilities which meets users' needs.

Acknowledgment. Authors received research funds from 59 the Basque Government as the head of the Grupo de Inteligencia Computacional, Universidad del Pais Vasco, UPV/EHU, from 2007 until 2025. The current code for the grant is IT1689-22. Additionally, authors participate in Elkartek projects KK-2022/00051 and KK-2021/00070. The Spanish MCIN 5has also granted the authors a research project under code PID2020-116346GB-I00.

References

1. Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Yousafzai, A., Xia, F.: A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. Netw. Comput. Appl. **52**, 11–25 (2015)
2. Almutairi, J., Aldossary, M.: A novel approach for IoT tasks offloading in edge-cloud environments. J. Cloud Comput. **10**(1), 1–19 (2021)
3. 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, pp. 13–16 (2012)
4. Cong, P., Zhou, J., Li, L., Cao, K., Wei, T., Li, K.: A survey of hierarchical energy optimization for mobile edge computing: a perspective from end devices to the cloud. ACM Comput. Surv. (CSUR) **53**(2), 1–44 (2020)
5. Elgendy, I.A., Zhang, W.Z., Liu, C.Y., Hsu, C.H.: An efficient and secured framework for mobile cloud computing. IEEE Trans. Cloud Comput. **9**(1), 79–87 (2018)
6. Elgendy, I.A., Zhang, W., Tian, Y.C., Li, K.: Resource allocation and computation offloading with data security for mobile edge computing. Future Gener. Comput. Syst. **100**, 531–541 (2019)
7. Ferreto, T.C., Netto, M.A., Calheiros, R.N., De Rose, C.A.: Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. **27**(8), 1027–1034 (2011)
8. He, Q., et al.: A game-theoretical approach for user allocation in edge computing environment. IEEE Trans. Parallel Distrib. Syst. **31**(3), 515–529 (2019)
9. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. J. Netw. Syst. Manage. **23**, 567–619 (2015)
10. Lai, P., et al.: Optimal edge user allocation in edge computing with variable sized vector bin packing. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 230–245. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_15
11. Lai, P., et al.: Edge user allocation with dynamic quality of service. In: Yangui, S., Bouassida Rodriguez, I., Drira, K., Tari, Z. (eds.) ICSOC 2019. LNCS, vol. 11895, pp. 86–101. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33702-5_8
12. Lai, P., et al.: Cost-effective app user allocation in an edge computing environment. IEEE Trans. Cloud Comput. **10**(3), 1701–1713 (2020)

13. Peng, Q., et al.: Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In: 2019 IEEE International Conference on Web Services (ICWS), pp. 91–98. IEEE (2019)
14. Rababah, B., Alam, T., Eskicioglu, R.: The next generation internet of things architecture towards distributed intelligence: Reviews, applications, and research challenges. J. Telecommun. Electr. Comput. Eng. (JTEC) **12**(2) (2020)
15. Sahni, Y., Cao, J., Zhang, S., Yang, L.: Edge mesh: a new paradigm to enable distributed intelligence in internet of things. IEEE Access **5**, 16441–16458 (2017)
16. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
17. Tyagi, H., Kumar, R.: Cloud computing for IoT. Internet Things (IoT) Concepts Appl., 25–41 (2020)
18. Yousefpour, A., et al.: All one needs to know about fog computing and related edge computing paradigms: a complete survey. J. Syst. Archit. **98**, 289–330 (2019)