sensors

MDPI

Article

Optimal Resource Provisioning and Task Offloading for Network-Aware and Federated Edge Computing

Avilia Kusumaputeri Nugroho [1], Shigeo Shioda [2] and Taewoon Kim [1,*]

[1] School of Computer Science and Engineering, Pusan National University, Busan 46241, Republic of Korea; avilia22@pusan.ac.kr
[2] Graduate School of Engineering, Chiba University, Inage-ku, Chiba 263-8522, Japan; shioda@faculty.chiba-u.jp
* Correspondence: taewoon@pusan.ac.kr

Abstract: Compared to cloud computing, mobile edge computing (MEC) is a promising solution for delay-sensitive applications due to its proximity to end users. Because of its ability to offload resource-intensive tasks to nearby edge servers, MEC allows a diverse range of compute- and storage-intensive applications to operate on resource-constrained devices. The optimal utilization of MEC can lead to enhanced responsiveness and quality of service, but it requires careful design from the perspective of user-base station association, virtualized resource provisioning, and task distribution. Also, considering the limited exploration of the federation concept in the existing literature, its impacts on the allocation and management of resources still remain not widely recognized. In this paper, we study the network and MEC resource scheduling problem, where some edge servers are federated, limiting resource expansion within the same federations. The integration of network and MEC is crucial, emphasizing the necessity of a joint approach. In this work, we present NAFEOS, a proposed solution formulated as a two-stage algorithm that can effectively integrate association optimization with vertical and horizontal scaling. The Stage-1 problem optimizes the user-base station association and federation assignment so that the edge servers can be utilized in a balanced manner. The following Stage-2 dynamically schedules both vertical and horizontal scaling so that the fluctuating task-offloading demands from users are fulfilled. The extensive evaluations and comparison results show that the proposed approach can effectively achieve optimal resource utilization.

Keywords: mobile edge computing; task offloading; optimal association; vertical scaling; horizontal scaling

check for updates

Citation: Nugroho, A.K.; Shioda, S.; Kim, T. Optimal Resource Provisioning and Task Offloading for Network-Aware and Federated Edge Computing. *Sensors* **2023**, *23*, 9200. https://doi.org/10.3390/s23229200

Academic Editors: Younghan Kim, Ngoc-Thanh Dinh and Min Wei

Received: 14 October 2023
Revised: 9 November 2023
Accepted: 14 November 2023
Published: 15 November 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, the growing number of delay-sensitive applications operating on resource-constrained devices has presented major challenges to the efficient execution of these applications. In particular, the emergence of the Internet of Things (IoT) has highlighted the need for efficient and responsive computing solutions [1]. To address these challenges, mobile edge computing (MEC) has emerged, which is regarded as a distributed version of cloud computing. MEC has been widely studied in depth [2–5] as a promising solution to offloading resource-intensive tasks to edge servers located in the vicinity of the user devices [6–8]. In particular, the common goals are reducing network latency [9], enhancing responsiveness [10], and improving quality of service (QoS) [11]. By utilizing the rich computation and storage resources of edge servers, MEC can accelerate the performance of applications running on user devices. In addition, it conserves the limited resources of end-devices, thereby preserving their operational lifetime.

However, the effectiveness of MEC depends on a well-designed resource scheduling strategy that makes optimum utilization of the resources that are related to each other. In particular, the optimal utilization of MEC required two key aspects to be addressed: the association of users with base stations (BS), the provisioning of virtualized resources on

edge servers (ES), and the dynamic distribution of tasks across edge servers. Achieving an optimal balance among these factors is of significance, as it can result in reduced response time and battery consumption at user devices. User-edge server association optimization [12] is a fundamental aspect of MEC optimization that ensures the balanced utilization of edge servers. For example, the balanced distribution of users across edge servers can prevent over- or under-provisioning problems. This not only enhances the QoS for individual users but also maximizes the efficiency of the MEC system as a whole. Achieving such balanced user-edge server associations requires careful consideration of factors such as workload distribution [13].

Due to its importance, various MEC optimization approaches have been explored [14–17]. However, the optimal use of MEC involves several challenges from various aspects. Many optimization methodologies currently in use have a limited scope, focusing on individual challenges within the MEC architecture. These optimization efforts frequently concentrate on individual elements such as edge server resource allocation, task-offloading strategies, and user-server assignments. Although these approaches provide valuable insights and opportunities for performance enhancement, their limits arise from their constrained scope. They often overlook considering comprehensive factors such as network resource assignments, vertical/horizontal resource provisioning, and integrated task offloading. These limitations emphasize the need for an approach that integrates all of these factors.

An optimal MEC approach should consider a wider range of factors, from the capabilities of individual edge servers to the network as well as the dynamic requirements of users and applications. In general, an edge server is located inside or connected via a direct link to a base station [18], and thus the association between user and base station can also determine the association between user and edge server. One key strength of edge computing is the capability of carrying out scaling in a dynamic manner. Optimized scaling can be performed either vertically or horizontally. The previous studies optimized them either independently instead of considering them together as an integrated approach to resource management. While several studies have explored the benefits of vertical scaling (VS) [19,20], horizontal scaling (HS) strategies have also garnered attention for their potential [21,22]. VS involves adjusting the capacity of individual edge servers by reallocating computational resources such as CPU, memory, and storage to handle varying workloads. HS, on the other hand, focuses on adding or removing edge servers to adapt to changing demands and maintain optimal system performance.

Despite the study conducted on each scaling approach, there is still a need for further research to enable both VS and HS. Expanding on this concept, our research emphasizes the potential of enabling both VS and HS in order to achieve enhanced resource utilization and QoS. By combining both scaling approaches, we create a dynamic environment in which not only the capacity of individual edge servers but also the number of edge servers is adjusted to handle varying workloads. This approach effectively utilizes the benefits of optimization, scaling, and responsiveness. Furthermore, joint consideration is also a crucial aspect of enhancing system efficiency. Network-side optimization ensures the efficient functioning of user-base station associations, while MEC optimization enhances the performance of resource allocation (both VS and HS) and task offloading. Although these aspects are often viewed as distinct notions, their integration has the potential to significantly reduce response times.

To further enhance the optimal MEC strategy, it should consider a wide range of factors, including the capabilities of individual edge servers, the effects of network resource scheduling, and dynamic user and application requirements. In addition, we consider the single-provider system of MEC, where a single service provider operates its own edge servers, each with different features and constraints. An organization or business can lease one or more edge servers, which can be exclusively used by the corresponding service subscribers. In such a scenario, a fully centralized optimization strategy is feasible for the single provider with its own regulations and requirements.

The concept of edge server federation represents a plausible use case, necessitating an examination of how the introduction of federation impacts dynamics and resource allocation. This prompts questions about resource management across federations, the feasibility of high-speed communication between federations, and the implications for user assignments and task offloading in a federated setup. Notably, the dynamics of resource allocation, load balancing, and task management may differ significantly when compared to a non-federated edge server environment.

This organizational structure of edge servers, organized into distinct federations, serves different purposes, whether for security reasons or to accommodate various organizations. The significance of this organizational structure increases in contexts with single providers. In instances where an individual provider operates multiple federations of edge servers, each of which caters to a distinct function, centralized optimization may be more applicable within each distinct federation. One additional factor to point out is that when edge servers are operated by different operators or when security is of important concern [23], horizontal scaling, or even the migrations [24] can be allowed only within a federation of edge servers.

Given the aforementioned considerations, our research methodology aims to provide an integrated optimization approach that effectively incorporates edge server provisioning by merging the principles of both VS and HS and task offloading assignment, alongside network resource scheduling. The comprehensive methodology holds potential for finding novel approaches to enhance performance and optimize resource utilization in a federated edge computing environment. The contributions of this paper are summarized as follows:

- In contrast to previous optimization approaches that focused on each individual component of the MEC architecture, we propose a comprehensive optimal approach that optimizes a chain of components in the MEC as well as network resources. In this study, the proposed approach optimizes BS-user association, federated-user assignment, resource provisioning, and task offloading.
- We propose a *federated* edge server-based MEC architecture, where a user assigned to a particular federation can utilize only the edge servers in the same federation. This is practical and essential when some edge servers are operated by different service provider, or when some edge servers are owned by a third-party organization that is not trustworthy.
- In contrast to the previous MEC optimization approaches that focused on either VS or HS for provisioning virtualized resources, we propose to enable both to further enhance resource utilization and users' QoS.
- We refer to our proposed approach as NAFEOS, which stands for Network-Aware Federated Edge Computing Optimal Scheduling. The NAFEOS approach presented in this study is formulated as a two-stage algorithm, considering the execution interval and complexity. The Stage-1 problem, which runs at long intervals, includes binary variables, resulting in relatively higher complexity. However, due to the efficient algorithms, such as branch-and-bound and branch-and-cut, leveraged in the computer solver we used in this study, the mixed integer (binary) linear program we propose in this study can be computed efficiently. On the other hand, the Stage-2 problem that iteratively optimizes both ES resources and task offloading at short intervals is formulated as linear programming so that it can run at low complexity. The proposed problem formulation aims to minimize battery consumption and service delay from the users' perspective. At the same time, it maximizes the fair load distribution among federations by having a multi-objective optimization solution.
- We have carried out extensive evaluations and validated the effectiveness of the proposed optimal approach. Also, we have performed a comparison study with the common approaches. To do so, we implemented the proposed method along with its variants.

The remainder of this paper is structured as follows: Section 2 summarizes the related studies. The following Section 3 presents the proposed two-staged approach that optimizes

network and edge server resources jointly. Evaluation, validation and analysis of the proposed solution are carried out in Section 4, and Section 5 concludes this paper.

2. Related Work

This section provides a comprehensive review of MEC and resource management literature. Prior studies have explored different aspects of MEC, often focusing on specific resource management components. User-BS associations have been an important part of MEC research. In line with our research, Wang et al. [25] proposed an optimization method for the association between users and base stations in MEC. This study prioritizes minimizing system delay and emphasizes efficient user-BS associations for enhancing QoS. The work, however, is limited in that it does not consider the scaling of virtualized resources, which is an essential factor in the optimal use of MEC.

In the context of user-edge server association in the context of MEC, Dai et al. [26] proposed a computational offloading framework that integrates both compute offloading and user-edge server association in a two-tier architecture. Tang et al. [27] introduced a task offloading approach that enhances the effectiveness of joint optimization techniques. The primary area of their research centers around optimizing tasks at the individual level. Despite their noteworthy findings related to the method of task offloading, the scope of this study was constrained to a single mobile device and a single mobile edge server. In addition, Bi et al. [28] proposed an integrated strategy for the joint optimization of computation-offloading decisions in MEC systems. Our method goes further by emphasizing the need for associations to enable low-latency communication within ES federations, facilitated by high-speed networks, all while considering federation-specific resource availability.

Edge server resource provisioning within MEC has also been studied. In [29], the study delved into MEC resource management within the Internet of Things (IoT) context, with a primary focus on optimizing resource efficiency and minimizing network costs. In other studies, ref. [30] proposed a comprehensive strategy concerning computing power allocation and efficient traffic scheduling for edge service provisioning. Furthermore, ref. [31] introduced the concept of resource provisioning in edge computing, with a special emphasis on applications demanding low-latency performance.

The exploration of federated edge computing has been conducted in various domains in several previous works [32–34]. Hussain et al. [32], introduced a federated edge computing approach for disaster management in remote smart oil fields, emphasizing resource allocation and load balancing for smart oil fields' robustness. Chi et al. [33] proposed DEEP-NET, a fully decentralized on-demand MEC-SC peer-offloading network that emphasizes QoS-aware load balancing, improved latency, and the protection of service providers' privacy. This approach leverages a federated gradient descent-based algorithm that operates in a fully decentralized manner. Karakoç et al. [34] proposed a Federated Edge Network Utility Maximization (FEdg-NUM) architecture, centered around clients with private utilities and communication within a peer-to-peer network of edge servers. While the aforementioned studies investigate federated edge computing across diverse domains, our research concentrates on network-aware resource provisioning and task offloading, encompassing a wider spectrum of edge servers and applications. Our work offers a comprehensive approach to resource optimization within the context of MEC, recognizing the significance of both MEC federating concepts and the federated edge computing paradigm.

In the optimization of MEC, the user-edge server association and the exploration of both VS and HS strategies have been studied as well. Regarding VS and HS, a cluster of notable research papers has been highlighted [35–38]. In [35], an innovative elastic edge cloud resource management model is proposed, which effectively combines the VS capability with HS. Expanding on this concept, ref. [36] proposes an adaptive auto-scaling technique for delay-sensitive serverless services. This method employs a complex combination of VS and HS that are intelligently tailored to the specific resource profiles of the services. Daraje et al. [37] presented a novel hybrid resource scaling strategy that stands out in the context of cloud computing. This method combines the capabilities

of VS and HS in an effort to optimize resource utilization. However, it is notable that none of these approaches address the integration of optimal user-BS/ES association and dynamic task distribution comprehensively. Maia et al. [38] addressed the critical issue of optimizing the location of scalable Internet of Things (IoT) services within the domain of edge computing. Their research examines both VS and HS, highlighting the importance of service deployment and scaling in edge computing.

Compared to the studies discussed earlier, our work introduces a novel joint optimization framework for the chain of resources addressed above. This integrated methodology provides the flexibility to process the user's offloading requests at either edge servers, the cloud, or even at the user's device, thus enhancing the ES-cloud offloading system's efficiency. Our work stands out for its capacity to handle these interdependent decisions collectively rather than separately. This work enables both VS and HS to further enhance the system's performance and users QoS. Considering the notion of edge server federations raises the practicality of the proposed approach. The optimization of user-BS association, which is the network-side resource, also plays an important role, especially in edge computing with mobile devices, and thus it cannot be excluded from the task of MEC optimization. Overall, the comprehensive optimization for MEC proposed in this work results in optimized resource utilization and enhanced performance within the MEC ecosystem.

3. Proposed Idea

In this section, we illustrate the proposed system architecture, and then introduce the proposed optimal MEC management scheme.

3.1. Proposed System Architecture

Figure 1 shows the overall system architecture we propose. The users (or user devices) at the bottom layer can connect to the network via access networks. Base stations (BS) at the access network can associate with users so that their offloading requests can be redirected to either ES(s) or the cloud. Each BS can associate and communicate with up to a certain number of users simultaneously, which is defined by the number of available orthogonal channels, N_{CH}. On top of the access network layer, ESs are deployed in federations. ESs belonging to the same federation can efficiently communicate with each other at low latency by being connected via a high-speed network. Such a system, for example, can be implemented by a software-defined network [39]. There is a one-to-one mapping between ES and BS, meaning that associating with a BS automatically determines which federation to belong to. To put it another way, to belong to a particular federation, a user should be associated with the BS that is connected to an ES in the federation. At the topmost layer is the cloud data center, which has enough computing resources, whereas the others, i.e., user devices and ESs, are resource-limited.

Offloading can be carried out in three different ways: self-offloading, offloading to ES(s), and offloading to the cloud. Users with enough computing resources on their devices can process their requests by themselves. Also, the user's request can be offloaded to one or more ESs or the cloud. If HS is supported, a user can offload its request to multiple ESs in the same federation. Finally, excessive amount of requests can be offloaded to the cloud. The delay between a user and BS is very small compared to the rest of the delays we consider in this paper. To be specific, it can be computed by dividing the BS-user distance, e.g., a few hundreds of meters, divided by the speed of light which is normally 3×10^8. The delay between ESs in the same federation is assumed to be short for being connected with each other by high-speed links. However, the links between ESs and the cloud are of large delay due to the large distance between ESs and the cloud. Users requests can be partitioned into fractional portions, and they can be processed in a distributed manner, possibly in different layers as well.

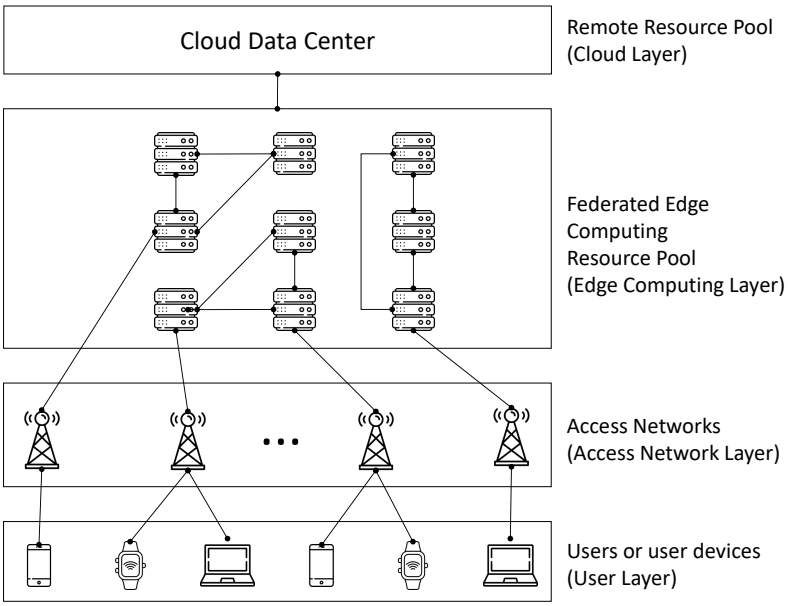

Figure 1. Overall system architecture consisting of four layers.

3.2. Assumptions

In this work, we make the following assumptions. The number of federation of ES, N_G, is known in advance, along with which ES belongs to which federation. Such a relation is abstracted by the federation indicator matrix $\mathbf{I}_{grp} \in \{0,1\}^{N_G \times N_S}$, where N_S, the number of ESs or BSs in the system, is known in advance. In the matrix, if the g-th row and s-th column are one, ES s belongs to federation g. Users are assumed to be stationary with their locations known. Given the locations of BSs, the accessibility matrix $\mathbf{I}_{acc} \in \{0,1\}^{N_S \times N_U}$ is constructed to indicate which user u can access (or within the transmission coverage of) which BS s, where N_U is the known number of users in the system. In the matrix, if the s-th row and u-th column are one, user u can access BS s. A user can associate with a BS only when the user can access the BS. It is assumed that the average amount of task offloading requests for each user per time unit is known by using the historic logs, and denoted by $\mathbf{r} \in [0,1]^{N_U \times 1}$. The computing resource budget available at users' devices, ESs and the cloud is denoted by $\mathbf{c}_{user} \in \mathcal{R}_{++}^{N_U}$, $\mathbf{c}_{es} \in \mathcal{R}_{++}^{N_S}$ and c_{cloud}, respectively, where $c_{cloud} \in \mathcal{R}_{++}$ is assumed to be a large number and $\mathcal{R}_{++}$ indicates a strongly positive real number.

3.3. Proposed Optimal MEC Management Method

In this work, we propose a joint optimization of user-BS association, ES provisioning via VS and HS, and task distribution so that users' requests can be processed efficiently. In contrast to the previous works focusing on each issue separately, we argue that such a chain of decisions should be considered at the same time to maximize the utilization of the ES-cloud offloading system. This is because one decision in a prior step can affect the following steps. Also, joint orchestration of VS and HS can further enhance the quality of service (QoS) of users as well as the resource utilization of the computing units (i.e., ES and cloud).

The proposed system consists of multiple layers as aforementioned, and there are multiple federations of ESs that are operated by different service providers. Associating a user with a BS leads to establishing a membership relation between the user and a federation as well. Thus, ill-considered association can yield undesired outcomes such as a certain federation being over-populated. Since a user can access the resource only within the same federation, it is not a desired situation. Also, BS can associate with up to a limited number of users, and thus an intelligent method for making an association and establishing a membership relation is required.

VS on ES can increase or decrease the allocated resources for the user's request, but due to the limited resources available on each ES, it may not suffice to fulfill the user's task-offloading demand. In such a case, allocating additional resources to other ES(s) is essential, called HS. Joint consideration of VS and HS can satisfy the user's QoS, especially when the user's demand is high or a certain ES is assigned to multiple users. Although the cloud resource pool is large enough, due to the increased delay when communicating with a remote cloud data center, it is desired to utilize as many ES resources as possible.

Considering the possible dynamic adjustment of ES resource allocation within a single ES or multiple ESs in the same federation, it is better to maintain enough amount of available, unused resources in each federation of ESs to be prepared for the possibility of upcoming offloading demand increase. In this paper, an effective load-balancing scheme among federations is proposed so that federations of ESs can process similar amounts of tasks and to secure enough amount of available resources therein.

Also, it is desired to use fewer resources on users' devices since they are battery-limited. The objective of the proposed method is to achieve load balancing among federations of ES, to maximize the lifetime of users devices by minimizing the amount of tasks processed locally at the users' devices, and to minimize the response time by minimizing the amount of tasks processed at the remote cloud data center.

The NAFEOS approach consists of two stages as shown in Figure 2: Stage-1: pre-configuration and Stage-2: real-time resource provisioning and task offloading. Stage-1 determines BS-user association and federated-user assignment. To make optimal decisions in the stage, Stage-1 also optimizes the resource provisioning and task offloading based on the average offloading requests from users. Once Stage-1 yields an optimal decision, the following Stage-2 iterates to make real-time optimal decisions regarding resource provisioning and task offloading upon receiving real-time task-offloading demand.

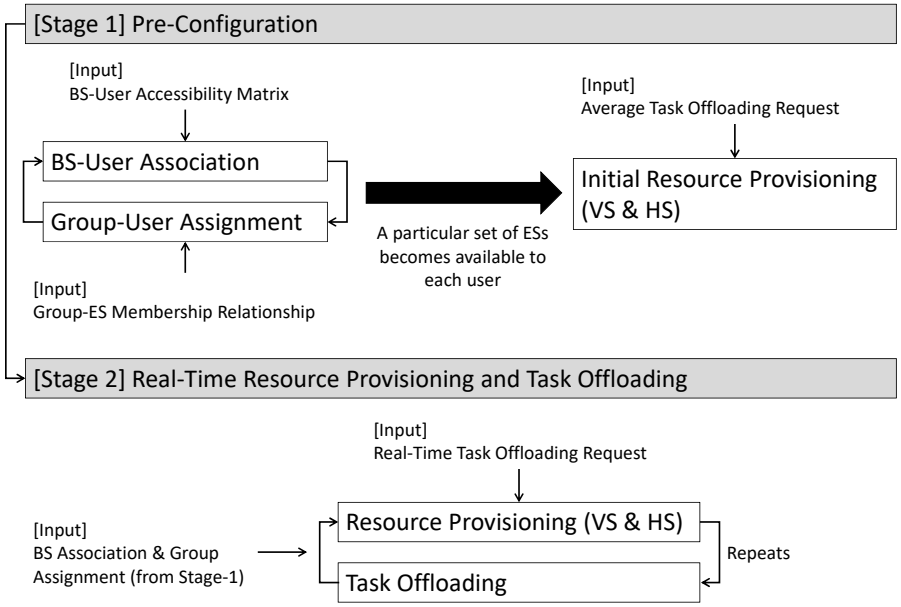

Figure 2. Overall flow of the NAFEOS method consisting of two stages.

The Stage-1 optimization problem in NAFEOS can be formally presented as follows. The objective (1) is to minimize the three terms with the given weights α, β, and $1 - \alpha - \beta$, where $\alpha + \beta \leq 1$. The two non-negative design parameters do not exceed the value of 1,

i.e., $\alpha, \beta \in [0,1]$, and the three strongly positive denominators s_1, s_2, and s_3 are used to scale the corresponding terms within the same range $[0,1]$.

$$\min_{\substack{\mathbf{A},\mathbf{G},\mathbf{Y},\\ \mathbf{x}_{user},\mathbf{X}_{es},\\ \mathbf{x}_{cloud},b}} \frac{\alpha}{s_1} b + \frac{\beta}{s_2} \mathbf{1}_{1 \cdot N_U} \times \mathbf{x}_{cloud} + \frac{1-\alpha-\beta}{s_3} \mathbf{1}_{1 \times N_U} \cdot \mathbf{x}_{user} \tag{1}$$

The first term in (1) minimizes b with which the offloaded load among federations can be balanced due to the constraint (15) to be addressed shortly. To be specific, the value of b is used to set the upper bound of the load across all federations and thus, minimizing b yields a fair load distribution. The second term minimizes the amount of task offloaded to the cloud (i.e., $\mathbf{x}_{cloud} \in [0,1]^{N_U \times 1}$), where the u-th element in $\mathbf{x}_{cloud}$ corresponds to the amount of user u's load offloaded to the cloud. The main purpose of the second term is to reduce the response time since the large distance between the user and the cloud yields a large network delay. The third term minimizes the amount of self-offloading $\mathbf{x}_{user} \in [0,1]^{N_U \times 1}$ (i.e., processing on the device itself), where the u-th element in $\mathbf{x}_{user}$ determines the amount of self-offloading for user u. This term plays an important role in reducing the power consumption of the user device and prolonging its lifetime. The $\mathbf{1}_{N \times 1}$ and $\mathbf{1}_{1 \times N}$ used in the objective function are a column and row vector of N number ones, respectively.

By minimizing the objective function, load balancing among federations can be guaranteed while both the service delay and battery consumption for users are minimized. To achieve the goal under practical considerations, we have defined the following constraints. The BS-user association decision is binary as shown below, called (2). The (s,u)-th element in $\mathbf{A}$ determines whether the BS s accepts the association request from user u or not by having the value be 1 or 0, respectively.

$$\mathbf{A} \in \{0,1\}^{N_S \times N_U} \tag{2}$$

In practice, the BS-user association can be made only when the user is placed within the transmission range of a BS. The following constraint (3) places an element-wise less-than or equal condition between $\mathbf{A}$ and $\mathbf{I}_{acc}$ with the $\preceq$ operator. The binary constant of the (s,u)-th element in $\mathbf{I}_{acc}$ corresponds to whether the user u can receive the pilot signal from the BS s or not by having the value of 1 or 0, respectively. Thus, the constraint (3) facilitates the BS-user association only when both can communicate with each other.

$$\mathbf{A} \preceq \mathbf{I}_{acc} \tag{3}$$

In this work, we assume a single antenna device for users and thus, each user can associate with a single BS at a time by the following constraint (4), where $\simeq$ is the element-wise equal operator.

$$\mathbf{A}^T \times \mathbf{1}_{N_S \times 1} \simeq \mathbf{1}_{N_U \times 1} \tag{4}$$

Each BS can associate with up to a particular number of users simultaneously, and the number is limited by the number of orthogonal channels, N_{CH}. Thus, the following constraint (5) is used to limit the number of users that a BS can allow network access to at a time.

$$\mathbf{A} \times \mathbf{1}_{N_U \times 1} \preceq N_{CH} \cdot \mathbf{1}_{N_S \times 1} \tag{5}$$

The federation-user mapping is also a binary relation as described in (6). That is, having the value of 1 for the (g,u)-the element in $\mathbf{G}$ indicates the federation g has decided to accept the user u so that the user can offload its processing load to the edge servers in the federation.

$$\mathbf{G} \in \{0,1\}^{N_G \times N_U} \tag{6}$$

In addition, one federation is exclusive of the rest by the assumption in this work, each user should become a member of a single federation by the constraint (7).

$$\mathbf{G}^T \times \mathbf{1}_{N_G \times 1} \simeq \mathbf{1}_{N_U \times 1} \tag{7}$$

A user can offload its task to an ES if the user is assigned with an isolated virtual environment on the ES (8). The (s, u)-th element in $\mathbf{Y}$ corresponds to whether the edge server s has allowed user u to offload its task or not, if the value is 1 or 0, respectively.

$$\mathbf{Y} \in \{0, 1\}^{N_S \times N_U} \tag{8}$$

In this study, we assume a containerized virtual environment such as Docker [40] which is light-weight and widely used in MEC [24]. Highly-loaded users may use multiple ESs for distributed task offloading, but the user can utilize only the ESs belonging to the same federation (9).

$$\mathbf{Y}^T \preceq \mathbf{G}^T \times \mathbf{I}_{grp} \tag{9}$$

The portion of task offloaded to the device itself, one or more ESs and the cloud is determined by the corresponding variables $\mathbf{x}_{user}$, $\mathbf{X}_{es}$ and $\mathbf{x}_{cloud}$, respectively (10).

$$\mathbf{x}_{user} \in [0, 1]^{N_U \times 1}, \mathbf{x}_{cloud} \in [0, 1]^{N_U \times 1}, \mathbf{X}_{es} \in [0, 1]^{N_S \times N_U}, \tag{10}$$

While the cloud is assumed to have enough resources to allow any amount of task offloading, both user devices and edge servers are of limited capacities as shown in (11) and (12), respectively.

$$\mathbf{x}_{user} \preceq \mathbf{c}_{user} \tag{11}$$

$$\mathbf{X}_{es} \times \mathbf{1}_{N_U \times 1} + h \cdot \mathbf{Y} \times \mathbf{1}_{N_U \times 1} \preceq \mathbf{c}_{es} \tag{12}$$

Each u-th element and s-th element in $\mathbf{c}_{user}$ and $\mathbf{c}_{es}$ corresponds to the computing resource budget of user u and edge server s, respectively. To calculate the amount of resource in use for each BS, we also consider the overhead to run virtual containers, represented by h.

A user can offload its task to one or more ESs if there is a container dedicated to the user in the corresponding edge server as described in the constraint (13).

$$\mathbf{X}_{es} \preceq \mathbf{Y} \tag{13}$$

Each user's QoS should be fully satisfied by the constraint (14), and it is assumed to be always possible due to the abundant resource in the cloud.

$$\mathbf{x}_{user} + \mathbf{x}_{es}^T \times \mathbf{1}_{N_S \times 1} + \mathbf{x}_{cloud} \simeq \mathbf{r} \tag{14}$$

The last constraint (15) sets the upper-bound b for the workload offloaded to federations, which is used to achieve load balancing.

$$(\mathbf{I}_{grp} \times \mathbf{X}_{es}) \times \mathbf{1}_{N_U \times 1} \preceq b \cdot \mathbf{1}_{N_G \times 1} \tag{15}$$

Putting it all together, we have the following Stage-1 optimization problem (called P. 16).

$$\text{(P. 16)} \min_{\substack{\mathbf{A},\mathbf{G},\mathbf{Y},\\ \mathbf{x}_{user},\mathbf{X}_{es},\\ \mathbf{x}_{cloud},b}} \frac{\alpha}{s_1} b + \frac{\beta}{s_2} \mathbf{1}_{1\times N_U} \times \mathbf{x}_{cloud} + \frac{1-\alpha-\beta}{s_3} \mathbf{1}_{1\times N_U} \times \mathbf{x}_{user}$$
$$\text{subject to (2)–(15)}$$

Given the optimal solutions $\mathbf{A}^*$, $\mathbf{G}^*$ and $\mathbf{Y}^*$ from the Stage-1 problem (P. 16), Stage-2 iteratively makes decisions on resource provisioning (i.e., VS and HS) and task offloading upon receiving real-time offloading requests. The Stage-2 optimization problem (called P. 17) is a subset of P. 16, and it is formally defined as below, where $\gamma \in [0, 1]$ is a design parameter, indicating the weight given to minimizing the use of the remote cloud resource.

$$\text{(P. 17)} \min_{\substack{\mathbf{x}_{user},\mathbf{X}_{es},\\ \mathbf{x}_{cloud}}} \frac{\gamma}{s_2} \mathbf{1}_{1\times N_U} \times \mathbf{x}_{cloud} + \frac{1-\gamma}{s_3} \mathbf{1}_{1\times N_U} \times \mathbf{x}_{user}$$
$$\text{s.t. (10), (11), (12), (13), (14)}$$

The proposed problem P. 16 is non-convex due to the binary (or integer in general) variables $\mathbf{A}$ and $\mathbf{G}$. However, due to efficient algorithms such as branch-and-bound and branch-and-cut [41], the given problem can be efficiently solved by computer solvers such as CPLEX [42] and Gurobi [43]. Also, P. 16 is an off-line method that can run often, meaning that the computation complexity is of less importance. On the other hand, P. 17 is for a real-time, iterative algorithm that should run each time slot. Due to the linearity of problem P. 17, its time complexity is polynomial [44] and thus, it is applicable to be used as a real-time scheduling algorithm. In spite of the presence of binary variables (or integers, in general) in the proposed problem formulation, one can efficiently find the global optimal solution with the widely used computer solvers, such as Gurobi and CPLEX. For example, to solve mixed integer programming-type problems, Gurobi which is used in this work employs branch-and-bound and branch-and-cut methods which are widely used exact solutions [45].

4. Evaluation

In this section, we first present the parameters assumed and used in our evaluation, along with the layout of the BS/ES and users. Also, the various algorithms adopted for performance comparisons are enumerated. The simulation and evaluation is carried out on a high-performance workstation with an Intel Core i9 10940X CPU and 128 GB memory, and the reported values in this section are the average out of ten evaluations. The evaluations were carried out on two different network configurations, namely a grid network and random network presented in Section 4.1 and Section 4.2, respectively.

4.1. Even Distribution of Base Stations

Figure 3 illustrates the assumed area for evaluation where 20 users (or user devices) and 25 BSs are evenly deployed. Users are placed at uniform random, while BSs are located at intersections on a grid. Each BS can communicate with users within 150 m radius coverage, and there are five orthogonal channels available so that up to five users are served simultaneously by a single BS. As shown in Figure 1, over the access network, there are 25 ESs federated into three groups and the ESs within the same federation are connected with high-speed communication links. We assume that the three federations of ESs are operated by different service providers, and thus HS, if supported, can occur with the ESs in the same federation but inter-federation HS is prohibited.

In our evaluation, we have abstracted both the users' offloading requests and the computing capacities such that they are denoted by a unit-less number in the range of $[0, 1.0]$. The rate at which the offloading request of each user is generated per unit of time is

randomly chosen from Uniform[0.20,1.00]. The resource budget of each user and ES for each unit time is randomly drawn from Uniform[0.05,0.20] and Uniform[0.40,0.80], respectively, indicating the random background processing workload. The cloud is assumed to have large enough resources to handle any amount of request. A user's request can be processed by the user's device, one or more ESs in the same federation, and/or the cloud. To provide the offloading service to the user, the ES shall create a light-weight container which consumes $h = 0.05$ amount of resource. The assumed parameters for evaluation are summarized in Table 1. The weights, α and β, are configured to 0.12 and 0.44, respectively, and the particular values are found by a heuristic approach.

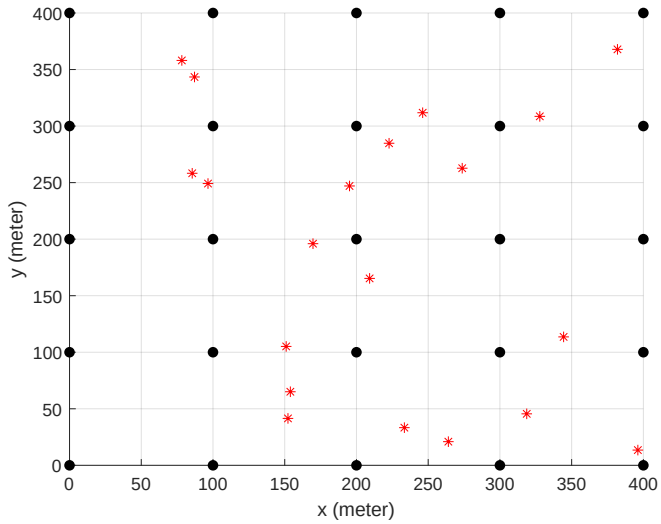

Figure 3. The layout of the assumed 400 m -by-400 m area where the 20 red stars and 25 black dots are the locations of users and BSs, respectively.

Table 1. Parameters used for evaluation on a grid network.

Parameter	Value
Number of users	20, distributed randomly
Number of BS/ES	25, distributed evenly
BS transmission range	150 m
Number of orthogonal channels	5
Number of federations	3
Offloading request rate	Uniform[0.20,1.00] per user
Resource budget	Uniform[0.05,0.20] per user Uniform[0.40,0.80] per ES
h	0.05 (container operating overhead)
weights	$\alpha = 0.12, \beta = 0.44$

We have implemented the NAFEOS and the simulation environment on MATLAB [46]. To solve the proposed optimization problem, we have used CVX [47] and Gurobi. For comparison, we have also implemented the following algorithms:

- NAFEOS: the optimal method proposed in this paper.
- RND (Random): random approach that makes decisions at random.
- RAG (Random Association and Grouping): same as NAFEOS, except that RAG randomly makes association and user-federation mapping (also called grouping).
- noHS (no Horizontal Scaling): same as NAFEOS, except that HS is not supported.

- noLB (no Load Balancing): same as NAFEOS, except that load balancing among federations is not supported.

Figure 4 shows the amount of processing units that are handled locally at the users' device. Except RND which randomly makes decisions, the rest of the algorithms perform optimal provisioning and task offloading. To be specific, NAFEOS, RAG, noHS, and noLB share a similar objective function that penalizes the use of users' devices for processing. As a result, RND partially lets users process their own requests locally, whereas the other algorithms do not. The reason why it is penalized in this work is to save battery consumption on the users' devices and to prolong their lifetime.

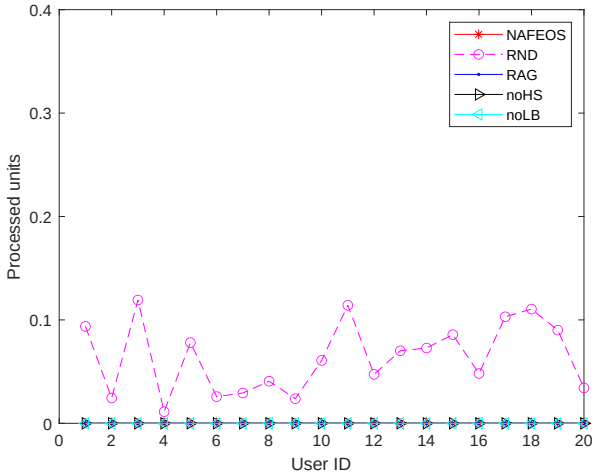

Figure 4. The amount of users' requests processed locally at the users' device on a grid network.

Figures 5 and 6 depict the average amount of processing units allocated at the ES and cloud, respectively, for each user's task offloading. As it can be seen from both figures, the NAFEOS, RAG and noLB can provision the optimal amount of processing units by using HS, and thus, there is no offloading to the cloud. Although noHS is another optimal provisioning scheme, it does not perform HS. If a single ES is assigned to multiple users and their aggregate request exceeds the ES's budget, the overflowing requests will be forwarded to the cloud. The downside of offloading to the cloud is the increased response time or end-to-end delay shown in Figure 7. With the assumption of a 5G cellular network as the underlying infrastructure [48], the achieved delay lends support to their relevance for various applications.

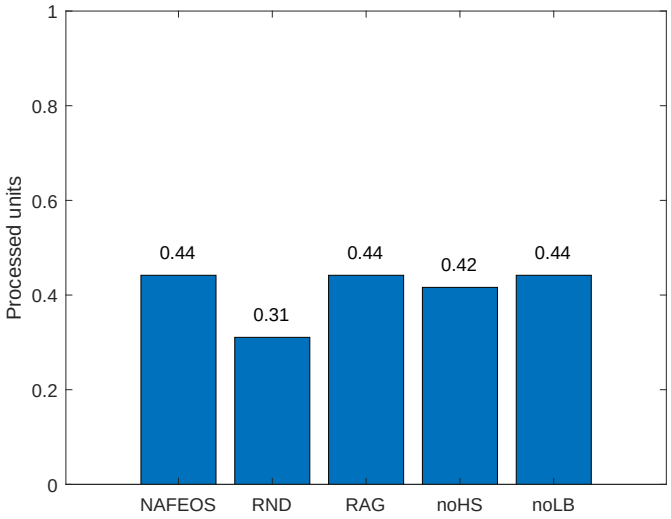

Figure 5. The average amount of users' requests processed at the edge server on a grid network.

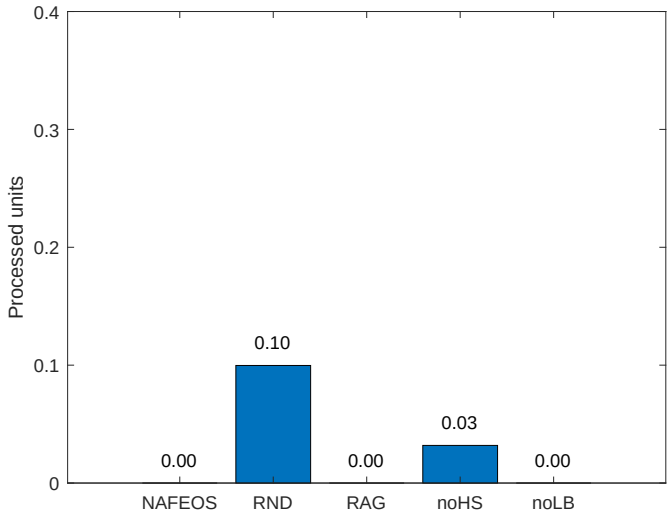

Figure 6. The average amount of users' requests processed at the remote cloud data center on a grid network.

For our evaluation, edge-to-edge and edge-to-cloud delays are set to 1.5 ms and 15 ms, respectively [49]. Given the distance between each user and its associated BS which can be computed by using their locations, the propagation delay between the two can be computed by dividing the distance by the speed of light. As it is already discussed in Section 3.1, we assume that the BS-user delay is negligibly small compared to the rest, edge-to-edge delay within the same federation is small, and edge-to-cloud delay is the largest in this study. Since the transmission range of a BS is up to 150 m, dividing the worst-case user-BS distance by the speed of light (i.e., 3×10^8 m/s) yields 50 µs, which is much smaller than the edge-to-edge delay. To be specific, the response time is computed as follows. For each time slot, for the proposed NAFEOS and other approaches that are considered for comparison, solve the corresponding algorithm to make decisions on self-offloading, ES offloading, and cloud offloading. Once the decision is carried out, the response time can be computed. Let d_{BS}, d_{e2e} and d_{e2c} be the one-way delay for BS-user, edge-to-edge, and edge-to-cloud. Then, for the following cases, the response time which excludes the time taken to process the offloaded task is computed as below.

- self-offloading yields zero response time
- offloading to ES without HS yields $2d_{BS}$
- offloading to ES with HS yields $2(d_{BS} + d_{e2e})$
- offloading to the cloud yields $2(d_{BS} + d_{e2c})$
- offloading to ES without HS and to the cloud yields $2(d_{BS} + d_{e2c})$
- offloading to ES with HS and to the cloud yields $2(d_{BS} + d_{e2e} + d_{e2c})$.

It is worth mentioning that the reported response time in this section is per-user average value, meaning that the summation of all response times is divided by the number of users reported in this section.

By using the above delay configurations, we can compute the average per-user response time (i.e., twice the end-to-end delay) as shown in Figure 7.

As it can be seen from the figure, the three optimal methods, i.e., NAFEOS, RAG, and noLB, outperformed the rest. The main reason for such low response time is because they do not offload to the cloud, which is causing the largest delay. In addition, due to the balanced load among federations, it is less likely that a certain ES is highly overloaded for the NAFEOS scheme. As a result, HS, which is causing an additional delay for the transmission among ES, is also minimized. Thus, the NAFEOS scheme achieved the lowest response time, although the improvement compared to RAG and noLB is insignificant. Both RND and noHS offload users' requests to the remote cloud which increased the response

time significantly. Among the two, RND achieved slightly better performance because it randomly lets users process their own tasks which is causing zero network delay.

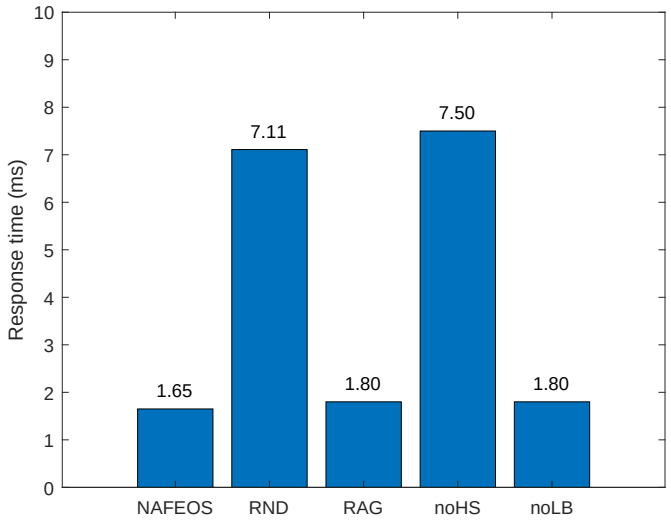

Figure 7. The average per-user offloading service response time ignoring the task processing time on a grid network.

The following figures, Figures 8–10, show the load-balancing performance among the three federations. The three optimal schemes, i.e., NAFEOS, RAG, and noLB, offload the entire task to ESs, and thus they achieve the largest offloaded units as shown in Figure 9 on average. However, due to the ill-considered association/federating and missing load-balancing features in RAG and noLB, respectively, their fairness performances are degraded as shown in Figure 10. To measure the load-balancing performance among federations, we have used the widely used Jain's fairness index [50] which measures the fairness as follows:

$$\mathcal{J}(x_1, x_2, \cdots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}.$$

Although noHS achieved almost perfect fairness, due to the inability to perform HS, it has offloaded a certain amount of task to the cloud, yielding lower performance than NAFEOS as shown in both Figures 8 and 9. Due to the random deployment of users, RND was able to achieve high fairness performance, but still it is outperformed by the NAFEOS.

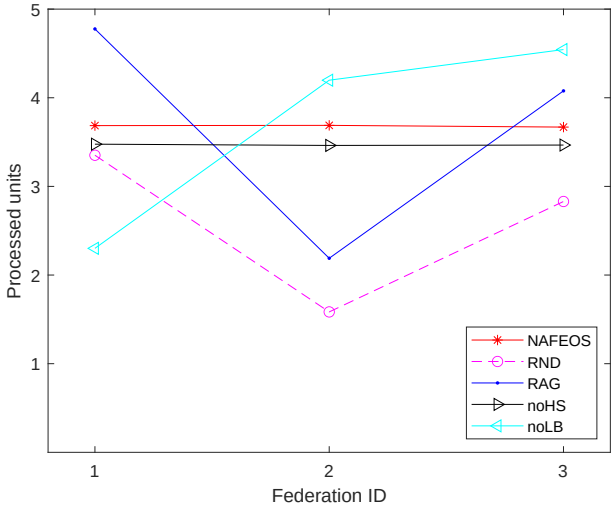

Figure 8. The average amount of processed units at each federation on a grid network.

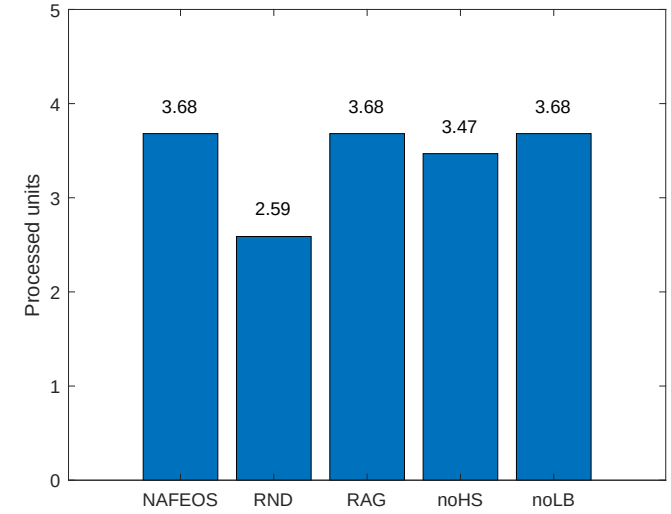

Figure 9. The average amount of processed units per federation on a grid network.

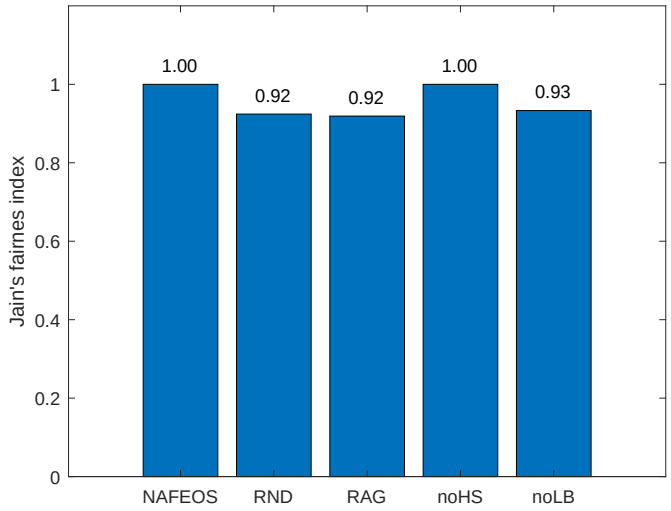

Figure 10. The performance of the fair distribution of the processed units among federations on a grid network.

4.2. Random Distribution of Base Stations with Higher Task Generation Rate

An additional evaluation has been carried out on a randomly located BS (see Figure 11) with a higher task generation rate. Table 2 summarizes only the parameters that are different from the previous ones in Table 1, and the weight parameters are heuristically chosen as before.

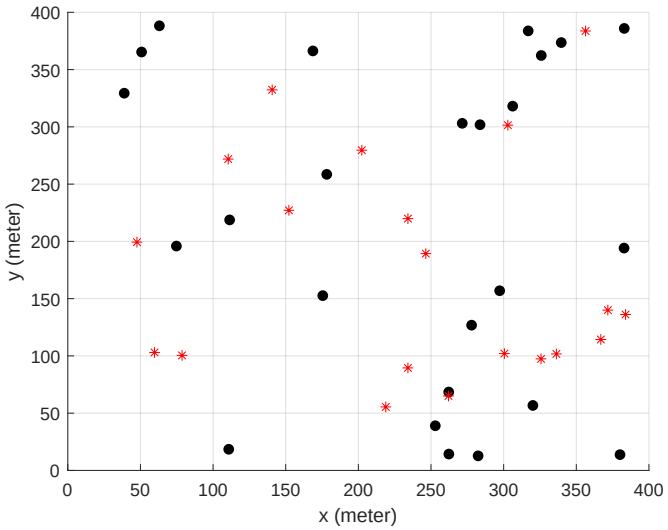

Figure 11. The layout of the assumed 400 m-by-400 m area where the 20 red stars and 25 black dots are the locations of users and BSs, respectively, that are randomly distributed.

Table 2. Changed parameters for evaluation on a random network.

Parameter	Value
Number of BS/ES	25, randomly distributed
Offloading request rate	$Uniform[0.60, 1.00]$ per user
weights	$\alpha = 0.22, \beta = 0.48$

Figure 12 shows the amount of task processed by the device itself. Except for noLB, all approaches let users process a small amount of load by themselves. This is quite different from the results from the evenly distributed BSs scenario with a moderate task generation rate in Figure 4. When each user is surrounded by a number of ESs and the amount of load to process is moderate, users do not offload tasks to themselves to minimize their battery consumption as shown in Figure 4. However, on the assumed network in this section where ESs are randomly deployed and the load generation rate is high, some users may not secure enough resources on ESs, and thus to reduce the service delay, devices process a small mount of tasks by themselves. On the other hand, noLB does not have restrictions on the even distribution of the load among federations and thus, it utilizes as much resource on ES as possible, resulting in no self-offloading on average.

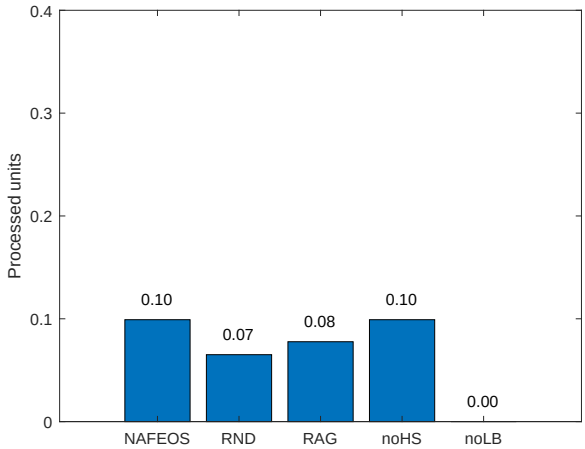

Figure 12. The amount of users' requests processed locally at the users' device on a random network.

Figures 13 and 14 show the amount of tasks offloaded to the ES and cloud, respectively. Except RND which randomly offloads the load, the rest of the approaches utilize a lot of resources from ES and a few from the cloud. The main reason for this is that utilizing ES yields a shorter delay. However, due to the limited resources on the ES, a small portion of the load is processed on the cloud anyway. The proposed NAFEOS offloads the least amount of load to the cloud, which effectively reduces the response time as shown in Figure 15. One interesting result here is that although RND offloads more to the cloud compared to noHS, its response time is much less than noHS. After analysis, what we found is as follows. In RND, a small number of users offloaded a lot of load to the cloud. On the other hand, noHS lets many users offload a small amount of task to the cloud. The noHS is not allowed to perform horizontal scaling. Thus, once the assigned ES is operating at its full capacity, users redirect the remaining load to the cloud.

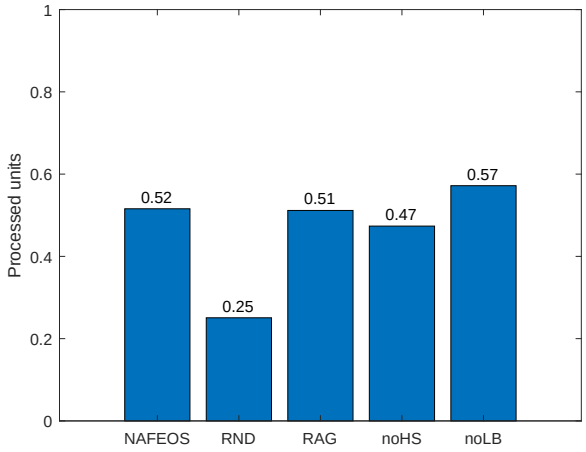

Figure 13. The average amount of users' request processed at edge server on a random network.

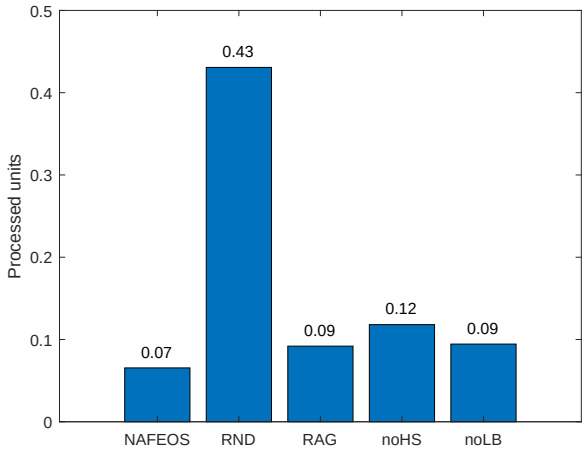

Figure 14. The average amount of users' request processed at the remote cloud data center on a random network.

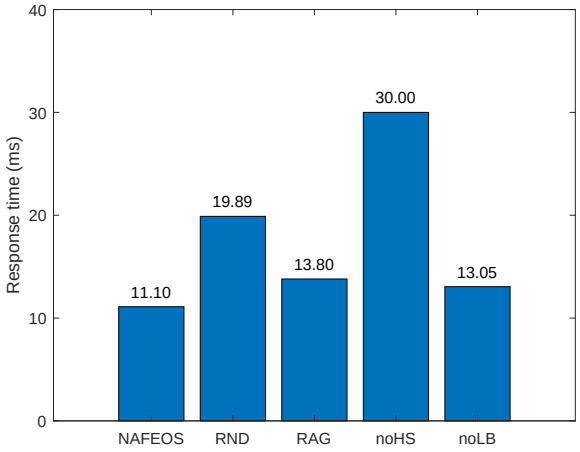

Figure 15. The average per-user offloading service response time ignoring the task processing time on a random network.

Figures 16–18 show the load balancing-related performance among federations. From both Figures 16 and 18, it is clear that the proposed NAFEOS achieves the highest load-balancing performance. Although noHS has achieved comparable fairness performance to NAFEOS, the amount of load offloaded to ESs is less than that of NAFEOS as shown in Figure 17. That is, the proposed NAFEOS can not only achieve high load-balancing performance, but also utilize as many resources on ES as possible which is an effective approach to reduce both the service delay and the battery consumption at the user device. Although noLB maximizes the use of ES resources, due to the lack of the load-balancing feature, its fairness performance is lower than that of NAFEOS.

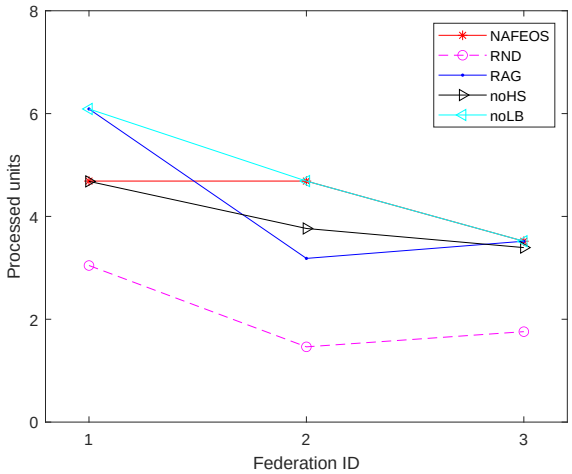

Figure 16. The average amount of processed units at each federation on a random network.

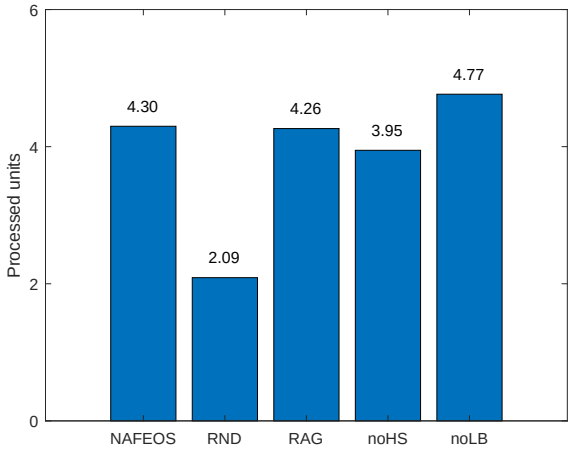

Figure 17. The average amount of processed units per federation on a random network.

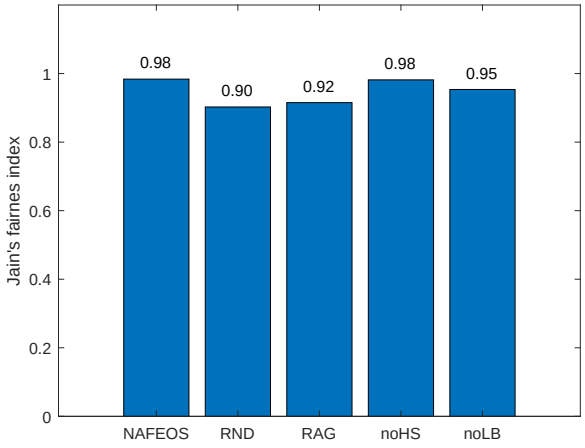

Figure 18. The performance of the fair distribution of the processed units among federations on a random network.

5. Conclusions

In this paper, we have introduced NAFEOS, an approach for optimal resource scheduling and task distribution in edge computing. In the NAFEOS system architecture, edge servers are federated, forming exclusive sets of available edge servers. A user's association with a base station determines which edge server and federation the user can access. Additionally, task offloading for users can be facilitated through horizontal and vertical scaling to meet the users' Quality of Service (QoS) requirements.

NAFEOS implements a two-stage approach, where the first Stage-1 solves the long-term decisions on the association between base station and user and the federation assignment between edge server federation and user along with initial provisioning of edge server resources. The following Stage-2 algorithm is repeatedly invoked on short time scales (or time slots) to make optimal decisions on edge server resource provisions by means of vertical and horizontal scaling. Then, it distributes the users' offloading requests to different layers.

The performance of NAFEOS, as well as several comparison algorithms, was systematically evaluated in two distinct network scenarios. The first scenario involved an even distribution of base stations (BS), while the second scenario involved a random distribution of BS with a higher task generation rate. In a grid network characterized by uniform distribution of BS and users, NAFEOS demonstrates outstanding performance in processing user requests at the ES. NAFEOS achieves a processing rate of 0.44 processing units, outperforming RND, RAG, noHS, and noLB. NAFEOS efficiently allocates processing units to the cloud,

thus reducing the demand for offloading and minimizing cloud utilization. It demonstrates optimal provisioning with a significantly lower per-user offloading service response time of 1.65 ms. NAFEOS shows its outstanding efficiency by achieving a processing unit count of 3.68 per federation. NAFEOS' outstanding load balancing and fairness among federations, confirmed by Jain's fairness index, establish NAFEOS as the optimal choice. In a random network with a higher task generation rate, NAFEOS shows outstanding results in task processing across different components. Specifically, it achieves a processing capacity of 0.10 units at users' devices, 0.52 units at the edge server, and effectively minimizes cloud offloading with a processing capacity of 0.07 units. The observed results show a lower average per-user offloading service response, with a value of 11.10 ms compared to other algorithms. NAFEOS also shows outstanding load-balancing capabilities, outperforming other algorithms with an average of 0.98 processed units per federation. The evaluation results show that NAFEOS is an effective approach for enhancing the efficiency and overall performance of IoT networks through association and federation, horizontal and vertical scaling, and workload balancing among federations.

NAFEOS has the potential to enhance network efficiency, extend user device lifetimes, and reduce response times in various low-powered Internet of Things applications such as remote monitoring of assets, smart surveillance systems, predictive maintenance in manufacturing, content caching and monitoring, and smart city/home. In particular, in the real-world applications operated mainly by low-power devices with network resource limits, the proposed NAFEOS is expected to provide the best efficiency gain. While the specific benefits may vary, the underlying principles of resource optimization remain consistent. In future work, we plan to carry out an empirical evaluation and integrate the federated learning into NAFEOS, further enhancing its ability to process user requests efficiently.

Author Contributions: Conceptualization, S.S. and T.K.; methodology, A.K.N. and T.K.; software, A.K.N. and T.K.; validation, S.S. and T.K.; formal analysis, S.S. and T.K.; investigation, A.K.N.; resources, T.K.; data curation, A.K.N.; writing—original draft preparation, A.K.N. and T.K.; writing—review and editing, S.S. and T.K.; visualization, A.K.N.; supervision, T.K.; project administration, T.K.; funding acquisition, T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1059109), and a New Faculty Research Grant of Pusan National University, 2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Y.; Ma, X.; Zhang, J.; Hossain, M.S.; Muhammad, G.; Amin, S.U. Edge Intelligence in the Cognitive Internet of Things: Improving Sensitivity and Interactivity. *IEEE Netw.* **2019**, *33*, 58–64. [CrossRef]
2. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [CrossRef]
3. Choi, P.; Kwak, J. A Survey on Mobile Edge Computing for Deep Learning. In Proceedings of the International Conference on Information Networking (ICOIN), Bangkok, Thailand, 11–14 January 2023; pp. 652–655.
4. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Internet Things J.* **2017**, *5*, 450–465. [CrossRef]
5. Shakarami, A.; Ghobaei-Arani, M.; Shahidinejad, A. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Comput. Netw.* **2020**, *182*, 107496. [CrossRef]
6. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
7. Jeong, H.J.; Lee, H.J.; Shin, K.Y.; Yoo, Y.H.; Moon, S.M. PerDNN: Offloading deep neural network computations to pervasive edge servers. In Proceedings of the IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 1055–1066.

8. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [CrossRef]
9. Liu, J.; Zhang, Q. Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications. *IEEE Access* **2018**, *6*, 12825–12837. [CrossRef]
10. Jiang, C.; Cheng, X.; Gao, H.; Zhou, X.; Wan, J. Toward Computation Offloading in Edge Computing: A Survey. *IEEE Access* **2019**, *7*, 131543–131558. [CrossRef]
11. Li, Q.; Wang, S.; Zhou, A.; Ma, X.; Yang, F.; Liu, A.X. QoS Driven Task Offloading with Statistical Guarantee in Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 278–290. [CrossRef]
12. Sardellitti, S.; Merluzzi, M.; Barbarossa, S. Optimal Association of Mobile Users to Multi-Access Edge Computing Resources. In Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
13. Zhang, P.; Zhang, A.; Xu, G. Optimized task distribution based on task requirements and time delay in edge computing environments. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103774. [CrossRef]
14. Xu, Y.; Zhang, T.; Liu, Y.; Yang, D.; Xiao, L.; Tao, M. Cellular-Connected Multi-UAV MEC Networks: An Online Stochastic Optimization Approach. *IEEE Trans. Commun.* **2022**, *70*, 6630–6647. [CrossRef]
15. Haibeh, L.A.; Yagoub, M.C.E.; Jarray, A. A Survey on Mobile Edge Computing Infrastructure: Design, Resource Management, and Optimization Approaches. *IEEE Access* **2022**, *10*, 27591–27610. [CrossRef]
16. Yang, J.; Shah, A.A.; Pezaros, D. A Survey of Energy Optimization Approaches for Computational Task Offloading and Resource Allocation in MEC Networks. *Electronics* **2023**, *12*, 3548. [CrossRef]
17. Chu, W.; Jia, X.; Yu, Z.; Lui, J.C.; Lin, Y. Joint Service Caching, Resource Allocation and Task Offloading for MEC-based Networks: A Multi-Layer Optimization Approach. *IEEE Trans. Mob. Comput.* **2023**, 1–17. [CrossRef]
18. Kim, T.; Lin, J.W.; Hsieh, C.T. Delay and QoS aware low complex optimal service provisioning for edge computing. *IEEE Trans. Veh. Technol.* **2023**, *72*, 1169–1183. [CrossRef]
19. Yahya, W.; Oki, E.; Lin, Y.D.; Lai, Y.C. Scaling and offloading optimization in pre-CORD and post-CORD multi-access edge computing. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4503–4516. [CrossRef]
20. Wang, N.; Matthaiou, M.; Nikolopoulos, D.S.; Varghese, B. DYVERSE: Dynamic vertical scaling in multi-tenant edge environments. *Future Gener. Comput. Syst.* **2020**, *108*, 598–612. [CrossRef]
21. da Silva, T.P.; Neto, A.F.R.; Batista, T.V.; Lopes, F.A.; Delicato, F.C.; Pires, P.F. Horizontal auto-scaling in edge computing environment using online machine learning. In Proceedings of the IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), AB, Canada, 25–28 October 2021, pp. 161–168.
22. Cañete, A.; Djemame, K.; Amor, M.; Fuentes, L.; Aljulayfi, A. A proactive energy-aware auto-scaling solution for edge-based infrastructures. In Proceedings of the IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), Vancouver, WA, USA, 6–9 December 2022; pp. 240–247.
23. Zhang, L.; Zou, Y.; Wang, W.; Jin, Z.; Su, Y.; Chen, H. Resource allocation and trust computing for blockchain-enabled edge computing system. *Comput. Secur.* **2021**, *105*, 102249. [CrossRef]
24. Kim, T.; Al-Tarazi, M.; Lin, J.W.; Choi, W. Optimal container migration for mobile edge computing: Algorithm, system design and implementation. *IEEE Access* **2021**, *9*, 158074–158090. [CrossRef]
25. Wang, H.; Wang, Y.; Sun, R.; Su, R.; Liu, B. Joint user association and power allocation for minimizing multi-bitrate video transmission delay in mobile-edge computing networks. In Proceedings of the 12th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2018), Sydney, NSW, Australia, 3–5 July 2019; pp. 467–478.
26. Dai, Y.; Xu, D.; Maharjan, S.; Zhang, Y. Joint computation offloading and user association in multi-task mobile edge computing. *IEEE Trans. Veh. Technol.* **2018**, *67*, 12313–12325. [CrossRef]
27. Tang, X.; Wen, Z.; Chen, J.; Li, Y.; Li, W. Joint optimization task offloading strategy for mobile edge computing. In Proceedings of the IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 17–19 December 2021; Volume 2, pp. 515–518.
28. Bi, S.; Huang, L.; Zhang, Y.J.A. Joint optimization of service caching placement and computation offloading in mobile edge computing systems. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 4947–4963. [CrossRef]
29. Kherraf, N.; Alameddine, H.A.; Sharafeddine, S.; Assi, C.M.; Ghrayeb, A. Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 459–474. [CrossRef]
30. Xiang, Z.; Deng, S.; Jiang, F.; Gao, H.; Tehari, J.; Yin, J. Computing power allocation and traffic scheduling for edge service provisioning. In Proceedings of the IEEE International Conference on Web Services (ICWS), Beijing, China, 19–23 October 2020; pp. 394–403.
31. Abouaomar, A.; Cherkaoui, S.; Mlika, Z.; Kobbane, A. Resource provisioning in edge computing for latency-sensitive applications. *IEEE Internet Things J.* **2021**, *8*, 11088–11099. [CrossRef]
32. Hussain, R.F.; Salehi, M.A.; Kovalenko, A.; Feng, Y.; Semiari, O. Federated edge computing for disaster management in remote smart oil fields. In Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 10–12 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 929–936.

33. Chi, H.R.; Radwan, A. Fully-Decentralized Fairness-Aware Federated MEC Small-Cell Peer-Offloading for Enterprise Management Networks. *IEEE Trans. Ind. Informatics* **2022**, *19*, 644–652. [CrossRef]
34. Karakoç, N.; Scaglione, A.; Reisslein, M.; Wu, R. Federated edge network utility maximization for a multi-server system: Algorithm and convergence. *IEEE/Acm Trans. Netw.* **2022**, *30*, 2002–2017. [CrossRef]
35. Li, C.; Tang, J.; Luo, Y. Elastic edge cloud resource management based on horizontal and vertical scaling. *J. Supercomput.* **2020**, *76*, 7707–7732. [CrossRef]
36. Zhang, Z.; Wang, T.; Li, A.; Zhang, W. Adaptive auto-scaling of delay-sensitive serverless services with reinforcement learning. In Proceedings of the IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 27 June–1 July 2022; pp. 866–871.
37. Daraje, M.; Shaikh, J. Hybrid resource scaling for dynamic workload in cloud computing. In Proceedings of the IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, India, 3–4 December 2021; pp. 1–6.
38. Maia, A.M.; Ghamri-Doudane, Y.; Vieira, D.; de Castro, M.F. Optimized Placement of Scalable IoT Services in Edge Computing. In Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 8–12 April 2019; pp. 189–197.
39. Li, C.; Qianqian, C.; Luo, Y. Low-latency edge cooperation caching based on base station cooperation in SDN based MEC. *Expert Syst. Appl.* **2022**, *191*, 116252. [CrossRef]
40. Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2.
41. Mitchell, J.E. Branch-and-cut algorithms for combinatorial optimization problems. *Handb. Appl. Optim.* **2002**, *1*, 65–77.
42. Cplex, IBM ILOG. V12. 1: User's Manual for CPLEX. *Int. Bus. Mach. Corp.* **2009**, *46*, 157.
43. Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*; Gurobi Optimization, LLC: Beaverton, OR, USA, 2023.
44. Karmarkar, N. A new polynomial-time algorithm for linear programming. In Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, Washington, DC, USA, 30 April–2 May 1984; pp. 302–311.
45. Morrison, D.R.; Jacobson, S.H.; Sauppe, J.J.; Sewell, E.C. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discret. Optim.* **2016**, *19*, 79–102. [CrossRef]
46. MATLAB. *Version 7.10.0 (R2010a)*; The MathWorks Inc.: Natick, MA, USA, 2010.
47. Grant, M.; Boyd, S. CVX: Matlab Software for Disciplined Convex Programming, Version 2.1, 2014.
48. Kelechi, A.H.; Alsharif, M.H.; Ramly, A.M.; Abdullah, N.F.; Nordin, R. The four-C framework for high capacity ultra-low latency in 5G networks: A review. *Energies* **2019**, *12*, 3449. [CrossRef]
49. Guo, M.; Li, L.; Guan, Q. Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems. *IEEE Access* **2019**, *7*, 78685–78697. [CrossRef]
50. Jain, R.K.; Chiu, D.M.W.; Hawe, W.R. *A Quantitative Measure of Fairness and Discrimination*; Eastern Research Laboratory, Digital Equipment Corporation: Hudson, MA, USA, 1984; Volume 21.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.