information

Article

Efficient Schemes for Optimizing Load Balancing and Communication Cost in Edge Computing Networks †

Efthymios Oikonomou ‡ and Angelos Rouskas *,‡

Department of Digital Systems, University of Piraeus, 18532 Piraeus, Greece; oikonomouef@unipi.gr
* Correspondence: arouskas@unipi.gr
† This article is a revised and expanded version of a paper entitled Selection of Service Nodes in Edge Computing Environments, which was presented at 2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Paris, France, 14–16 December 2020.
‡ Current address: M. Karaoli & A. Dimitriou 80, 18534 Piraeus, Greece.

Abstract: Edge computing architectures promise increased quality of service with low communication delays by bringing cloud services closer to the end-users, at the distributed edge servers of the network edge. Hosting server capabilities at access nodes, thereby yielding edge service nodes, offers service proximity to users and provides QoS guarantees. However, the placement of edge servers should match the level of demand for computing resources and the location of user load. Thus, it is necessary to devise schemes that select the most appropriate access nodes to host computing services and associate every remaining access node with the most proper service node to ensure optimal service delivery. In this paper, we formulate this problem as an optimization problem with a bi-objective function that aims at both communication cost minimization and load balance optimization. We propose schemes that tackle this problem and compare their performance against previously proposed heuristics that have been also adapted to target both optimization goals. We study how these algorithms behave in lattice and random grid network topologies with uniform and non-uniform workloads. The results validate the efficiency of our proposed schemes in addition to the significantly lower execution times compared to the other heuristics.

Keywords: edge computing; service nodes; access nodes; communication cost; load balancing; computational times

Citation: Oikonomou, E.; Rouskas, A. Efficient Schemes for Optimizing Load Balancing and Communication Cost in Edge Computing Networks. *Information* **2024**, *15*, 670. https://doi.org/10.3390/info15110670

Academic Editor: Aneta Poniszewska-Maranda

Received: 24 August 2024
Revised: 16 October 2024
Accepted: 23 October 2024
Published: 25 October 2024

1. Introduction

It is widely acknowledged that resource-intensive and time-sensitive applications cannot be adequately accommodated by the centralized cloud computing paradigm. These types of demanding modern applications and innovative technologies, such as online gaming, high-definition video streaming, Augmented Reality (AR) and vehicular communication technology (Internet of Vehicles—IoV) [1,2], rely heavily on high bandwidth and low end-to-end communication delay. In response to this challenge, the concept of edge computing has emerged, including various edge platform strategies as outlined in [3–7].

Edge computing [8,9], is a network architecture that facilitates cloud computing capabilities and IT services at the edge of a wireless access network. This design is intended to be implemented at the network edge servers (small-scale servers also known as edge nodes), allowing the execution of applications and the associated processing to occur closer to the customer which in turn leads to enhanced application performance, reduced service response time and decreased network and internet congestion. In addition, this proximity to users enables mobile operators to provide context-aware services and improve QoE by reducing network delay using radio access network information [10].

The edge service node placement or deployment is a key issue in edge computing research community. The edge service node can be deployed at any access node type in the

network, e.g., base transceiver station (BTS), WiFi hotspot, etc., supporting end-users' mobility and fulfilling the physical proximity necessary for low latency communication [11,12]. However, it is neither cost-effective nor energy-efficient to keep every edge service node operational at all time, especially when traffic intensity is low. Obviously, the decision about where to place or deploy these nodes must consider several factors, such as network topology, operator policies and user computational demands. Additionally, the selection of edge service nodes in conjunction with the end-user allocation to obtain the service from the most proper service point is also a complicated issue considering factors such as available resources, equal balance of load, service quality requirements, and user preferences.

The objective of our research is to tackle the challenge of identifying the most suitable access nodes for hosting operational (acting also as) service nodes and ensuring that all users are allocated efficiently in a balanced manner, while simultaneously improving their Quality of Experience (QoE) in terms of response time. We first formulate two separate optimizations problems, each one targeting the two different goals of communication cost minimization and load balance optimization and then we combine these problems into one bi-objective optimization problem using a normalized weighted sum objective function. We then present (a) an Edge Service Node Selection scheme, which determines the most appropriate operational subset of service nodes and (b) an Access Node Allocation scheme, which allocates users of the access nodes to the previously selected service node subset so that both objectives are fulfilled. We also adapted previously proposed heuristics for the *K*-median clustering problem to also include load balance as a criterion in their decisions.

Our proposed schemes are evaluated through simulation experiments on different network topologies considering both uniform and non-uniform workload distributions. The results indicate that our schemes are highly efficient in optimizing load balance distribution without sacrificing performance on the latency communication cost of the system, compared to the other proposed heuristics.

The main contributions of this article are the following:

- A comprehensive survey of existing physical edge server placement solutions to determine edge server deployments.
- The formal formulations of a communication cost minimization subproblem, a load balancing optimization subproblem and a bi-objective optimization problem that aims to jointly optimize load balancing and mean latency scores in the network.
- The formal description of a novel service node selection scheme that attempts to evenly distribute service nodes across the edge network.
- The formal description of a novel access nodes allocation scheme that attempts to allocate access nodes to the most appropriate service node by simultaneously incorporating proximity-based and load balance decision criteria.
- The ability of the algorithms to tackle both uniform and non-uniform (random) load distribution.
- The adaptation of other previously proposed heuristics to include load balance in their decisions and their formal descriptions.
- The time complexity analysis of all schemes presented.
- Our proposed algorithms not only produce near optimum load balance and effective communication cost solutions, but also exhibit great scalability and significantly lower execution times compared to the other proposed heuristics.

The paper is structured as follows: Section 2 presents related research works. In Section 3, the system model is introduced, and the optimization problem is formally formulated. Section 4 presents the proposed policies and previous and properly adapted heuristics. Section 5 presents the experimental setup, along with the performance evaluation results. Finally, Section 6 concludes our study.

2. Related Work

Extensive and detailed surveys about edge computing can be found in [13–20]. Existing research works focus on the physical placement of cloudlets within a network and how

to allocate users to obtain service. The authors in [21] proposed an effective approach for placing K cloudlets in a wireless metropolitan area and assigning users to these cloudlets in such a way that the lowest possible average system response time is achieved. In [22,23], the authors approach the problem of placing edge servers in some strategic locations with the objective of workload-balanced edge servers and minimum access delay between users and edge servers, combining K-means and mixed-integer quadratic programming algorithms. However, in this work, edge server locations may be different from the base station locations and K-means clustering is applicable in contrast to ours where only K-median clustering is possible.

A genetic algorithm and local search algorithms were used in [24] to find an optimal edge server allocation strategy. In [25], the authors proposed a method for determining the optimal placement of edge servers, through the partitioning of the network into areas of smaller size to simplify the process of resources distribution and used the deep Q-network and Markov game to improve resource load balancing and reduce global latency, while optimizing global resource allocation. This approach is different to ours, as optimization per zone may end up with underutilized computational resources at some zones, and the authors propose the sharing of resources among adjacent zones to solve this problem.

The research in [26] presents a fast heuristic approach along with a distributed genetic algorithm for a QoS-aware load balancing problem among cloudlets in a WMAN to optimize mobile application performance. The authors in [27] introduced an online routing and load balancing algorithm with the objective of minimizing the average time taken to offload the tasks. This paper considers that the cloudlet locations are predetermined while our problem is to spot the most appropriate base stations to host the edge servers. The work in [28] proposes a fog computing-based method called energy-aware load balancing and scheduling (ELBS) involving a workload-dependent, on the fog node, model for energy consumption, an optimization function for load balancing within the manufacturing cluster and an improved particle swarm optimization algorithm to achieve the best possible solution. The research in [29] presents a strategy for placing cloudlets, on the basis of historical data records consisting of snapshots of the users' requests, in order to minimize communication latency. Furthermore, the authors in [30] address a capacity-constrained cloudlet placement problem and propose an approximation algorithm as a solution for placing k cloudlets in a WMAN to achieve minimum communication delay between the user and the cloudlet.

One of the main challenges for a service provider is how to determine the optimal placement of an appropriate number of service nodes within a network, as well as how to effectively allocate users' resource requirements to those nodes. The authors in [31] conducted a study on the energy-aware edge servers placement problem. They approached it as a multi-objective optimization problem and based on the particle swarm optimization to find a placement strategy for the edge servers that minimizes the total energy consumption, while maintaining acceptable access delay levels. In [32], the authors determine the placement of cloudlets from a pool of candidate locations, taking into account both the associated OPEX and CAPEX expenses, as well as the access delay from task offloading. For the cloudlet placement, the authors in [33], proposed the cost-aware cloudlet placement in mobile edge computing (CAPABLE) strategy, and developed a Lagrangian heuristic algorithm in order to achieve the sub-optimal solution for this NP-hard cloudlet placement problem.

The research in [34] focuses on investigating methods for task allocation and provisioning in a minimum subset of placed cloudlets. The PACK algorithm, introduced in [35], addresses the capacitated location–allocation problem for server placement by minimizing the distances between servers and their corresponding access points. The algorithm considers various parameters such as workload distribution, latencies, server co-location with access points, and prioritized locations. In [36], the authors introduced a load balancing scheme called LAB, by assigning IoT devices to the most suitable fog nodes and base stations, to minimize the latency of IoT data flows, considering both computing and

communication latencies. The paper assumes that every base station hosts a fog node which is different from the edge server placement problem that we consider in this work.

In [37], the authors proposed a mobile hybrid hierarchical P2P (MHP2P) model as a cloudlet solution for MEC, offering high scalability and stability. To address load distribution challenges with increasing MEC servers, they introduced inter- and intra-cluster load balancing schemes within a three-layer architecture to optimize performance and mitigate load imbalances. In [38], the authors examined cluster-based techniques, including *K*-means and density-based cluster methods, for determining edge server locations. While density-based methods prioritize reducing latency, the focus of their study emphasized the importance of workload balancing across servers and as a result, they proposed a modified *K*-means algorithm that integrates heuristic techniques, enhancing both server placement and point allocation based on centroids.

While existing methods, such as *K*-means clustering, allow for flexible edge server placement, they prove inadequate for fixed location settings. Our research addresses the critical challenge of simultaneously optimizing communication costs and load balancing in edge computing, particularly in environments where edge servers are constrained to specific, predefined locations, like access nodes or base stations. By formulating the problem as a bi-objective optimization task constituting of a *K*-median clustering subtask and a load-balancing subtask, our approach provides a robust solution tailored to these constraints. Additionally, our approach also enables edge service providers to better prioritize their operational objectives and select the optimal deployment strategies, ensuring improved cost efficiency and overall system performance.

More specifically, our work stands out from prior research works as we consider an access edge network and formulate an edge server placement and users' access nodes association optimization problem with a bi-objective function that targets both (a) minimizing communication costs between end users and edge nodes, and (b) balancing the offered workload among the selected edge service node subset. In our model, all base stations in the network are possible candidates for hosting edge server nodes, while the number of edge servers depends on the computational load and capacity of the servers. We develop very fast schemes that manage to distribute edge service nodes fairly across the network and assign users' access nodes to service nodes so that both objectives are simultaneously satisfied under uniform and non-uniform traffic loads.

In our previous studies [39,40], our goal was to determine the most suitable configuration to efficiently balance the load while minimizing communication latency and ensuring optimum resource utilization across the network. In addition, during dynamic system operation, when a preferred service node lacks sufficient capacity, our approach reallocates requests to other service nodes to increase system performance and avoid underutilized resources [40]. However, these works lack both the formal formulations of the communication cost minimization and load balancing optimization subproblems and the bi-objective optimization problem. In this paper we provide the formal descriptions of our enhanced schemes. along with other previously proposed heuristics for the *K*-median problem, adapted to include load balancing in their decisions. Additionally, we also present an analysis of the time complexity analysis of these methods.

3. System Model and Problem Statement

We assume a wireless network, providing edge services and computational power to the devices of the customers of a mobile network operator. In this network, a set of M inter-connected access nodes (ANs) is deployed. We model the network topology of ANs as a connected graph G(V, E), with network ANs $V = \{v_1, v_2, \ldots, v_M\}$ and network links E interconnecting the ANs. Every node is accessible through one or more hops by every other node of the graph.

The operator provides access to edge services by deploying service nodes (SNs), with each SN being hosted (colocated) at a corresponding AN. Each AN has the capacity to operate one colocated SN, but the actual number N of the operational SNs depends on the

computational demands (load/tasks) imposed on the system by its network users. Thus, in this way, the provider activates only as many servers as necessary to serve the offered load, achieving at the same time better energy consumption.

In general, the subset of operational SNs can be denoted as $S = \{s_1, s_2, \ldots, s_N\}$, $S \subseteq V$, $|S| = N$ and $N \leq M$. For convenience, we also define the mapping function $ind(s_i)$ which returns the index j of AN v_j that hosts SN s_i. We assume that the users of AN $v_{ind(s_i)}$ are served by SN s_i. Obviously, when $N = M$, $S = V$ and one SN is activated at every AN in the network. However, $N < M$ indicates that the users of those ANs, that do not have a colocated SN, should be served by a SN that operates at some other AN.

Let node weight w_j denote the user computational workload offered to AN v_j, which is proportional to the number of users accessing network and edge resources through v_j. With the special case $w_j = 1$, $j = 1, \ldots, M$, we assume that every AN serves the same number of end-users, which corresponds to a uniform distribution of users across the network area. Different values for w_j's correspond to non-uniform distribution of users.

For each AN v_j, the cost of communicating or accessing the edge cloud services at some SN s_i hosted by AN v_k can be represented by the path communication cost d_{jk}, which incorporates latency and capacity costs of the intermediate links of the minimum communication cost path between of v_j with v_k. Obviously, $d_{kk} = 0$. For the requirements of this work, we assume that the necessary capacity to support this communication is always available across the intermediate links of this path.

Finally, let C be the computational capacity of each SN, assuming that all SNs are similarly dimensioned.

As the management of the radio access network falls out of the scope of our paper, we make the assumption that all the wireless links between end-user devices and their corresponding wireless AN have the same wireless capacity and latency. Without loss of generality, we also assume that all interconnecting transmission links (edges) in the graph are identical to each other and share the same characteristics, specifically in terms of transmission capacity and latency; thus, the communication cost d_{jk} is fully determined by the number of links between v_j and v_k.

The assumed system model in this study, is shown in Figure 1. At the lowest layer there is a set of eight end-user devices, connected via wireless links to their closest AN, requesting edge resources and services. The network comprises a total of nine (9) ANs, where only five (5) of them hosting an operational service node. Consequently, the users from the remaining four (4) ANs, the colocated SN of which is not active, request edge resources and services at some of the five (5) operational SNs.

Given a specific set of ANs and the required number of SNs (related to the offered workload) in the system, the edge service node management module objective is to minimize the overall communication network expenses experienced by network users, while at the same time maintaining a fair (balanced) workload distribution among the SNs. When $N < M$, meaning there are fewer SNs than ANs, this formulation should select the most appropriate subset of ANs to host the SNs and allocate the ANs that do not host some SN to the most suitable SN, with the objective of (a) minimizing the network's response time for delivering edge cloud services and (b) achieving a balanced workload distribution among the SNs. The problem is considered as an optimization challenge with two distinct objectives. In the following, we start by formulating two separate problems, each one defined by a single objective function, and then we combine both problems into a single bi-objective function.

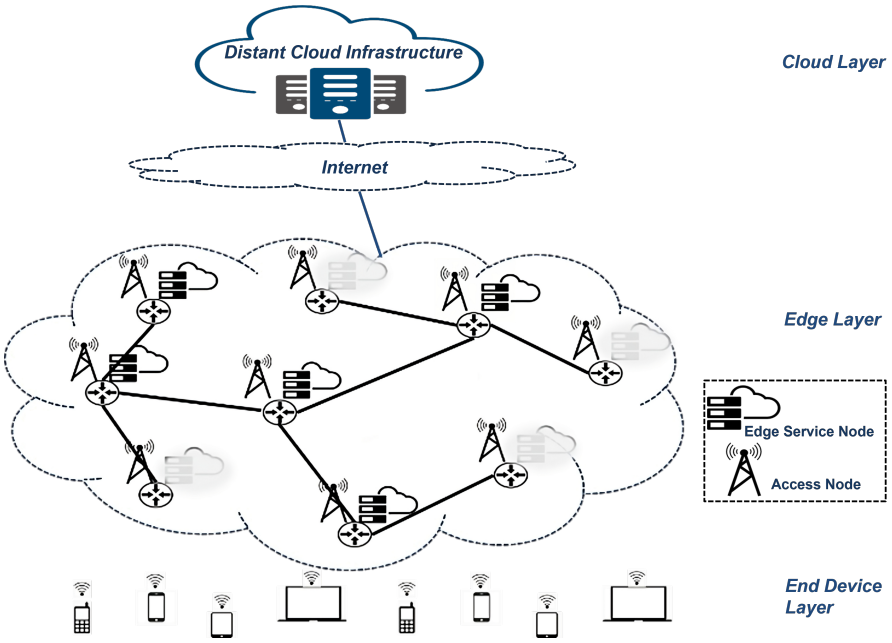


Figure 1. Edge computing network architecture: 9 Access Nodes with 5 operational Service Nodes assumed in total.

The problem of minimizing total response time for all users is equivalent to minimizing the total communication cost and can be formulated as an optimization problem as follows:

$$\min TotCommCost = \sum_{k=1}^{M} \sum_{j=1}^{M} w_j \cdot d_{jk} \cdot x_{jk} \tag{1a}$$

Subject to:

$$\sum_{k=1}^{M} x_{kk} = N \tag{1b}$$

$$\sum_{k=1}^{M} x_{jk} = 1, \quad j = 1, \ldots, M \tag{1c}$$

$$\sum_{j=1}^{M} w_j x_{jk} \leq C, \quad k = 1, \ldots, M \tag{1d}$$

$$x_{jk} \in \{0, 1\}, \quad j, k = 1, \ldots, M \tag{1e}$$

where the following binary decision variables are introduced:

$$x_{jk} = \begin{cases} 1, & \text{if AN } v_j \text{ is served by a SN instantiated at AN } v_k \\ 0, & \text{otherwise} \end{cases}, \quad j, k = 1, \ldots, M \tag{2}$$

which indicates whether AN v_j is served by the SN in AN v_k. Constraint (1b) ensures that exactly N SNs are activated, each SN at some AN, while constraints (1c) ensure that every AN v_j is served by exactly one SN, which is activated at some AN v_k. Constraints (1d) ensure that the computational load of all ANs served by some SN hosted at some AN v_k does not exceed the computational capacity of the SN. Finally, constraints (1e) and definition (2) ensure only integer solutions. This problem is also known as the capacitated K-median problem in clustering [41]. Given a set of nodes (or clients) L in a metric space, we seek K nodes from L that will act as cluster heads, so that the total sum of distances of clients from their cluster head is minimized, while ensuring the number of clients of each cluster head does not exceed the cluster head capacity.

The load balancing optimization formulation, on the other hand, should attempt to fairly balance the load across the SNs. A way to achieve this is by minimizing the load assigned of the most loaded SN. Formally, this can be expressed as follows:

$$\min \quad \max_{k\in\{1,\ldots,M\}} \sum_{j=1}^{M} w_j x_{jk} \tag{3}$$

Subject to the following: (1b)–(1e).

This problem can be reformulated in its standard epigraph form [42], as follows:

$$\min W \tag{4a}$$

Subject to (1b)–(1e) and

$$\sum_{j=1}^{M} w_j x_{jk} \leq W, \quad k = 1, \ldots, M \tag{4b}$$

Constraints (4b) in conjunction with the optimization objective enforce that the computational load of the most loaded SN is minimized. The minimum the load of the most loaded SN, the more balanced SNs' load will be.

Both problems (1) and (4), may not have feasible solutions if the requested computational resources of constraints (1d) cannot be satisfied by N SNs. In this case, one or even more SNs would be additionally required for the problems to have feasible solutions. An approach would be to increase N by one and formulate and solve new problems and so on, until the feasible solutions set is nonempty. Instead, we will follow a different approach and study equivalent problems, by relaxing capacity constraints (1d), and providing solutions for all possible values of N operational SNs, $N = 1, 2, \ldots$ Thus, assuming N SNs as operational, we solve the non-capacitated versions of the problems, e.g., without the capacity constraints (1d), and if the capacity of the most heavily loaded SN of the solution is not adequate to serve its allocated computational load, then $N + 1$ SNs are set as operational and the non-capacitated problems are reformulated and solved next. If again the capacity of the most heavily loaded SN is not adequate to serve its allocated computational load, the solution with $N + 2$ operational nodes is examined, and so on.

In a most realistic scenario, an edge service provider, even though the expected computational burden of ANs could be easily hosted by a small number of SNs, would prefer to operate a larger number of SNs so that the communication cost be kept at lower levels. Thus, it is often more useful to provide solutions with varying number of SNs and a fair balanced load distribution among them, for each solution, so that the edge network/service operator may decide the most proper operational set of SNs targeting both communication cost between SNs and their assigned ANs along with efficient and balanced utilization of SNs' computational resources.

These two conflicting optimization problems, having solutions of different scales, can be combined into one bi-objective optimization problem by use of the weighted sum method after normalizing both objectives in the (0,1) space to obtain a Pareto efficient solution [43]. For the minimum communication cost problem (1), we can first substitute the absolute objective function (1a) with the average communication cost by dividing with the sum of weights over all ANs and use the equivalent objective function (5) instead. Considering diameter D of the network, defined as the length of the shortest path between the most distanced nodes of the network, we can map the resulting average communication cost to the (0,1) space using D and zero as its upper and lower bounds, respectively.

$$\min \; AvgCommCost = \frac{1}{\sum_{j=1}^{M} w_j} \sum_{k=1}^{M} \sum_{j=1}^{M} w_j \cdot d_{jk} \cdot x_{jk} \tag{5}$$

For the load balancing optimization problem (4), we can easily devise a lower bound w_{min} of the objective function as the average load of the N SNs: $w_{min} = \frac{1}{N}\sum_{j=1}^{M} w_j$. An upper bound w_{max} of the objective function can be found by assuming that $N-1$ SNs are operated at the $N-1$ ANs with the smaller load weights w_j's and the rest $M-N+1$ ANs are assigned to the Nth SN which gets the sum of their load weights. This last SN is loaded with w_{max}, the maximum load possible.

Thus, the normalized weighted bi-objective problem is formulated as follows:

$$\min \lambda_{cc}\frac{AvgCommCost}{D} + \lambda_{lb}\frac{W - w_{min}}{w_{max} - w_{min}} \tag{6}$$

Subject to (1b), (1c), (1e) and (4b), where $\lambda_{cc} + \lambda_{lb} = 1$ and $0 < \lambda_{cc}, \lambda_{lb} < 1$.

Because the minimum communication cost problem is equivalent to the capacitated K-median clustering problem, which is NP-hard [44], the normalized bi-objective problem is also NP-hard. In the following, we will introduce computationally efficient heuristics to tackle this problem.

4. Proposed Policies

Assuming N SNs are operational to serve the computational requests of all ANs, we divide the weighted bi-objective problem into two separate submodules that are treated sequentially: (a) select N ANs that will act also as SNs (initiate the hosted/colocated SN) and (b) assign the rest $M-N$ ANs to the most appropriate SN. We will refer to these two submodules as the *Service Node Selection* submodule and *Access Node Allocation* to SNs submodule, respectively.

4.1. Edge Service Node Selection Scheme

Each AN in the network topology can host a SN. Given the network topology of M ANs and the number N of SNs, the proposed algorithm searches for the most suitable set S of N ANs, whose members would host a SN. The basic idea is to evenly distribute the SNs across ANs of the network so that communication costs from other ANs are relatively lower.

We first introduce a metric to grade each AN v_k on how costly it is for all other ANs to communicate with this node. We define $d_k = \sum_{j=1}^{M} w_j \cdot d_{jk}$, as the weighted sum of communication costs of every AN v_k in the network (Step 1). This metric gives an indication of the cost for the users of all ANs to communicate with AN v_k.

We then sort the list of ANs in increasing order of d_k's (Step 2). The first node in this sorted list Q has the minimum cost to communicate with all other ANs of the network. Let $v_{k^*} = arg\min_k d_k$ be the first node in Q and $d_{k^*}^{min} = \min_k d_k$.

In case of one operational SN, we simply select as SN the AN that presents the lowest weighted communication latency cost (Step 5). When the computational load is uniform, this is also the most "central" node, meaning that this node is located deep into the network, and as such, it has the minimum delay cost to communicate with *all* other ANs. However, when the computational load is non-uniform, intuitively the best SN is closest to the most highly loaded ANs in the network. If there are more than one candidate SNs with the same minimum value, we randomly select one of them.

When the procedure should select two or more SNs, let $d_{k^*}^{max} = \max_j d_{jk^*}$ be the maximum communication cost between every other AN v_j in V and v_{k^*}. Then, we search for the first SN to add to S, by traversing the Q list until we find the first node that is at least $\frac{d_{k^*}^{max}}{2}$ costly to communicate with AN v_{k^*}. Let v_l be that node, which is added as the first SN in S. Again, let $d_l^{max} = \max_j d_{jl}$, be the maximum communication cost of some AN from v_l. To find the second SN, we again search the Q list from the start to locate the AN that is at least $\frac{d_l^{max}}{2}$ away from the first node v_l and its weighted sum of communication costs is at least the same as that of the first SN (not higher). Let v_m be that node, which is added as the second SN in S (Steps 7–11).

The remaining SNs, third and so on, are determined as follows. We set dist $= \frac{d_l^{max}}{2}$ and search from the start of the Q list for a node whose communication cost is at least *dist* away from *all* the already selected service nodes (Steps 12–14). If no such node exists, then we decrease *dist* by one (Step 15) and repeat the search until a node is found. If such a node is found, then we select it as the third node, and so on. Otherwise, we continue to reduce the *dist* value by one, repeating the search procedure. If more than one such node exists, then we select the one whose sum of communication costs from the already selected SNs is minimum.

The time complexity of the Edge Service Node Selection implementation of Algorithm 1 is as follows. The time complexity of line 1 is $O(M^2)$ and dominates all lines up to line 11. If $|S| = p$, line 13 of the while loop requires the examination of $M - p$ candidates in the Q list, and for each candidate p distances are compared against *dist*, yielding $(M - p)p$ steps in total. However, if no AN fulfills the distance constraint, the search is repeated with $dist = dist - 1$ (line 15). The number of repetitions of line 15 is bounded by the diameter D of the network, defined as the length of the shortest path between the most distanced nodes of the network. In fact, the greater number of searches is encountered if all repetitions of line 15 are performed when $p = N - 1$, that is $(M - (N - 1))(N - 1)D$. Thus, the total number of repetitions is $\sum_{p=1}^{N-2}(M - p)p + (M - (N - 1))(N - 1)D$, which yields $O(MN^2 + MND)$, as the time complexity of the whole procedure. For small values of N, this becomes $O(MD)$, but for high values of N, the worst time complexity approaches $O(M^3)$.

Algorithm 1 Edge Service Node Selection

Input: Network topology $G(V, E)$ of M ANs, ANs' computational load $w_j, j = 1, \ldots, M$, and number N of SNs.
Output: Set S of N ANs to operate as edge SNs.

1. For each AN v_k calculate $d_k = \sum_{j=1}^{M} w_j \cdot d_{jk}$.
2. Set $Q = V$ and sort list Q in increasing order of d_k.
3. Set $S = \emptyset$.
4. If $N == 1$ {
5. Set $S = S \cup \{v_{k^*}\}$ and exit;
   }
6. If $N \geq 2$ {
7. Find the most distant AN $v_{k'}$ from v_{k^*}, i.e., the AN with the largest communication cost $d_{k'k^*}$.
8. Traverse Q until AN v_l is found, located at least $d_{k'k^*}/2$ away from v_{k^*}.
9. Find the most distant AN $v_{l'}$ from v_l, i.e., the AN with the largest communication cost $d_{l'l}$.
10. Traverse Q until AN v_m is found, located at least $dist = d_{l'l}/2$ away from v_l, $d_m \leq d_l$.
11. Set $S = S \cup \{v_l, v_m\}$.
    }
12. While $|S| < N$ {
13. Traverse Q until AN v_n is found, located at least *dist* away from every node in S. Ties are resolved by selecting the AN with the smallest sum of communication costs from all nodes in S.
14. Set $S = S \cup \{v_n\}$ and return.
15. If there is no such node set $dist = dist - 1$ and goto step 12.
    }
16. Exit;

4.2. Load Balanced and Node Proximity Access Node Allocation to SNs Scheme

Once we determine S, we then need to allocate each AN to SNs to deliver services to the end-users, attached to the corresponding AN. We propose a policy to be combined

with Edge Service Node Selection scheme in order to achieve both minimum network communication cost and balanced distribution of ANs to SNs.

The basic idea is that the procedure visits SNs in a round-robin (RR) manner, attempting to distribute the ANs among SNs in a balanced way, as much as possible. The formal description of the load balance assignment of ANs to SNs is shown in Algorithm 2. At first, each SN serves the users located within the same AN (Step 1). In every RR cycle all SNs are examined, however the order that SNs are visited may differ from cycle to cycle as will be shown. In addition to that, an AN is assigned to the SN under examination, only if the AN satisfies a certain criterion for this SN. The procedure ends when all ANs are assigned to some SN (Step 3).

Algorithm 2 Load Balanced and Node Proximity Access Node Allocation

Input: Network topology $G(V, E)$ of M ANs and set S of N ANs to operate as edge SNs.
Output: Set S_i with the ANs, assigned to SN s_i, $i = 1, \ldots, N$.

1. Initialize $S_i = \{v_{ind(s_i)}\}, i = 1, \ldots, N$
2. Calculate $d_{ji}^{norm}, j = 1, \ldots, M, i = 1, \ldots, N$
3. While there are ANs not yet assigned to some SN {
4. For each SN s_i
5. Find $d_{min,i}^{norm} = \min\limits_{v_j \text{ not assigned}} \{d_{ji}^{norm}\}$
6. Sort S in increasing order of $d_{min,i}^{norm}$
7. For $i = 1$ to N {
8. $d_{min,i}^{norm} = \min\limits_{v_j \text{ not assigned}} \{d_{ji}^{norm}\}$, $v_{j^*}^{(i)} = \arg \min\limits_{v_j \text{ not assigned}} (d_{ji}^{norm})$
9. Ties are resolved by selecting that $v_{j^*}^{(i)}$ which has also minimum absolute d_{ji}
10. if $d_{min,i}^{norm} \leq d_{equal}^{norm}$ {
11. $S_i = S_i \cup \{v_{j^*}^{(i)}\}$
12. Remove $v_{j^*}^{(i)}$ from the set of unassigned ANs
}
13. else skip this SN s_i
}
}
14. Exit;

For every AN v_j, we define the normalized relative communication cost of that AN from SN s_i as follows:

$$d_{ji}^{norm} = \frac{d_{j,ind(s_i)}}{\sum_{i=1}^{N} d_{j,ind(s_i)}}, \quad j = 1, \ldots, M, \quad i = 1, \ldots, N \tag{7}$$

This value represents the relative communication cost of AN v_j from SN s_i compared to the sum of communication costs of v_j from all SNs. If we assume, hypothetically, that an AN was equally distant from all SNs, in other words it has the same communication cost towards each SN, then the corresponding normalized distance would be $d_{equal}^{norm} = \frac{1}{N}$. This value can be considered as a threshold value to compare and realize whether an AN is closer to (farther from) a SN compared to some other SN, if its normalized relative distance value is greater (less) than d_{equal}^{norm}. Before the start of every RR cycle, every SN is characterized by metric $d_{min,i}^{norm} = \min\limits_{v_j \text{ not assigned}} \left\{d_{ji}^{norm}\right\}$, determined among the ANs not yet assigned to some SN (Step 4-5). SNs are then sorted in increasing order of this metric (Step 6) and this will be the order that SNs will be visited in this RR cycle. When an SN is visited, if its metric is less than d_{equal}^{norm} threshold (Step 10), the corresponding AN (Step 8) is

allocated to that SN (Step 11) and removed from further consideration (Step 12). If more than one AN has the same relative distance to this SN, the closest AN is selected (Step 9).

The time complexity of the Load Balanced and Node Proximity Access Node Allocation implementation of Algorithm 2 is as follows. Line 2 is $O(MN)$. The while loop is performed $M-N$ times because at the beginning N ANs are hosted by their colocated SN and $M-N$ ANs are unassigned. Let p be the number of ANs not assigned yet. The loop of line 4 requires Np searches, line 6 is $O(N \log N)$, and the loop of lines 7–13 also involves Np checks. Thus, the total number of searches is $\sum_{p=1}^{M-N} Np$, and the time complexity of the whole procedure is $O(M^2N)$, which for small values of N becomes $O(M^2)$, but for high values approaches $O(M^3)$.

4.3. Previously Proposed Approaches with Load Balance Enhancement

In this section, a set of previously proposed heuristics for the K-median problem are presented, namely, *Forward Greedy*, *Reverse Greedy* and a *Local Search* algorithm [45,46]. All of them are focusing on detecting the N most appropriate set of SNs so that the total sum of the communication cost from every AN to its *closest* SN is minimized. Thus, for a given set of N SNs S, the assignment of ANs to their *closest* SN always achieves minimum communication cost. In other words, if S_i is the set of ANs closest to SN s_i, $V = \bigcup_{i=1}^{N} S_i$, then $TCC(S) = \sum_{i=1}^{N} \sum_{v_j \in S_i} w_j \cdot d_{j,ind(s_i)}$ is minimum for this set S of SNs.

Nevertheless, as these heuristics are distance based, it is necessary to include load balancing in their decisions. If we let $L_i = \sum_{v_j \in S_i} w_j$, be the sum of weights of those ANs in S_i that corresponds to the computational load on SN s_i, a simple metric for measuring the balance of computational load among the SNs is the sample variance of L_i's with $\mu = \frac{1}{N} \sum_{j=1}^{M} w_j$ being the corresponding sample mean. Thus, for a given set S of N SNs and the corresponding assignment sets of ANs S_i's, a load balance metric for this set S and S_i's is given by Equation (8).

$$LB(S) = \frac{1}{N-1} \sum_{i=1}^{N} (L_i - \mu)^2 \tag{8}$$

4.3.1. Load Balanced Forward Greedy

The Forward Greedy algorithm starts with one SN in S, selecting as SN the AN that minimizes the weighted sum of communication costs of all ANs (Steps 1, 2). Then, at each step, the algorithm increases the number of SNs by one, always selecting as SN the v_k that will increase the communication cost by the least possible amount (Step 4). This is performed under the assumption that the ANs not in $S \cup \{v_k\}$ will be assigned to their closest (with least communication cost) SNs in $S \cup \{v_k\}$. The SN is added to S (Step 6) and the procedure is repeated until the number of SNs is N (Step 3). If more than one choice is possible, the algorithm bases its decisions on the load balance criterion and selects the SN v_k such that, when added, the ANs not in $S \cup \{v_k\}$ will be assigned to their closest SN and the assignment minimizes Equation (8) (Step 5).

The time complexity of the Forward Greedy implementation in Algorithm 3 is as follows. The complexity of line 1 is $O(M^2)$. In the while loop, let p be the number of SNs so far, thus $M-p$ ANs are possible candidates for selecting the next SN. Adding one SN results in $p+1$ SNs. Computing $TCC(S \cup \{v_k\})$ in line 4 requires checking $p+1$ distances and selecting the minimum one. This is performed for every AN, that is $M-(p+1)$ times, thus $TCC(S \cup \{v_k\})$ requires $(M-(p+1))(p+1)$ steps. To obtain the optimum value in line 4 requires checking all $M-p$ possible candidates; thus, the total complexity of line 4 is $O((M-p)(M-(p+1))(p+1))$, which is greater than the complexity $O((M-p)(p+1))$ of line 5. The number of while loops is $N-1$, which yields $\sum_{p=1}^{N-1}(M-p)(M-(p+1))(p+1)$ checks in total, and time complexity $O(M^2N^2)$ for the whole procedure. For low values of N, the complexity is $O(M^2)$, but for higher values it approaches $O(M^4)$.

4.3.2. Load Balanced Reverse Greedy

Unlike Forward Greedy, Reverse Greedy starts with M SNs by operating one SN at each AN (Step 1), and then at each step decreases the number of SNs by one, always selecting for removal the SN at AN v_k that will increase the communication cost by the least possible amount (Step 3) towards the remaining SNs, assuming that the ANs not in $S - \{v_k\}$ will be assigned to their closest (with least communication cost) SNs. The SN is removed (Step 5) and the procedure is repeated until the number of SNs is N (Step 2). If more than one choice is possible, then the algorithm bases its decisions on the load balance criterion and selects as SN the v_k that, when removed, the ANs not in $S - \{v_k\}$ will be assigned to their closest SN while also minimizing Equation (8) (Step 4).

Algorithm 3 Forward Greedy with Load Balance

Input: Network topology $G(V, E)$ of M ANs, ANs' computational load $w_j, j = 1, \ldots, M$, and number N of SNs.
Output: Set $S = \{s_i\}$ of N ANs to operate as edge SNs and sets S_i with the ANs, assigned to SN $s_i, i = 1, \ldots, N$.

1. Let $v_{k^*} = \arg\min_{v_k \in V} d_k$
2. Set $s_1 = v_{k^*}$; $S = \{s_1\}$
3. While $|S| < N$ {
4. $v_{k^*} = \arg\min_{v_k \in V - S} TCC(S \cup v_k)$
5. Ties are resolved by selecting v_{k^*}, which minimizes also Equation (8).
6. Set $S = S \cup \{v_{k^*}\}$
   }
7. Determine S_i's: Assign every AN v_j in $V - S$ to its closest SN s_i in S
8. Exit;

The time complexity of the Reverse Greedy implementation of Algorithm 4 is as follows. In the while loop let p be the number of SNs so far, thus p ANs are possible candidates for selecting the next SN to be removed. Removing one SN results in $p - 1$ SNs. As in Forward Greedy, computing $TCC(S - v_k)$ in line 3 requires $(M - (p - 1))(p - 1)$ steps, and the selection of the optimum value requires checking all p possible candidates; thus, the total complexity of line 3 is $O((M - (p - 1)(p - 1)p)$, which is greater than the complexity $O((p - 1)p)$ of line 4. The while loops are executed for $p = M, M - 1, \ldots, N + 1$, which yields $\sum_{p=N+1}^{M}(M - (p - 1))(p - 1)p$ checks in total, and time complexity $O(M^4)$ for the whole procedure.

Algorithm 4 Reverse Greedy with Load Balance

Input: Network topology $G(V, E)$ of M ANs, ANs' computational load $w_j, j = 1, \ldots, M$, and number N of SNs.
Output: Set $S = \{s_i\}$ of N ANs to operate as edge SNs and sets S_i with the ANs, assigned to SN $s_i, i = 1, \ldots, N$.

1. Set $S = V$
2. While $|S| > N$ {
3. $v_{k^*} = \arg\min_{v_k \in S} TCC(S - v_k)$
4. Ties are resolved by selecting v_{k^*}, which minimizes also Equation (8).
5. Set $S = S - \{v_{k^*}\}$
   }
6. Determine S_i's: Assign every AN v_j in $V - S$ to its closest SN s_i in S
7. Exit;

4.3.3. Load Balanced Local Search

In general, a Local Search procedure for the selection of SNs starts with an arbitrary initial set of N SNs and attempts to locate a new set of N SNs by swapping one or more SNs with some other ANs if a better communication cost is achieved. The procedure is repeated until no improvement in the communication cost is possible when swapping one or more SNs. The initial set of SNs can be arbitrary, random, or the outcome of another heuristic which can offer a good starting point for the Local Search.

An implementation of a Local Search procedure is given in Algorithm 5 with an additional characteristic of choosing solutions that yield better load balance than others in case of communication cost ties. The procedure starts with an initial set of SNs (Step 1), which can be a random choice or preferably a solution outcome of another heuristic, e.g., Forward Greedy. Once the assignment sets S_i's are determined (Step 2), the communication costs and load balance metrics are computed (Step 3) and are assumed as minimum. For each AN (loop of Step 4), the algorithm iterates through all SNs (loop of Step 6), and temporarily swaps them (Step 7). For each swap, it allocates ANs to SNs (Step 8) and calculates the total communication cost and load balance metrics. If either the communication cost is improved, or the communication cost remains unchanged, but the load balance metric is improved by this swap, the swap is noted along with the new minimum metric values (Step 9, 10). When all SNs are checked for the AN under examination and at least one swap occurred (Step 11), the swap with the best communication cost and best load balance metric becomes permanent (Step 12) and the procedure continues with the next AN, and so on. Thus, the procedure prioritizes communication cost, but when the cost remains the same, it considers load balance improvement.

Algorithm 5 Local Search with Load Balance

Input: Network topology $G(V, E)$ of M ANs, ANs' computational load $w_j, j = 1, \ldots, M$, and set $S_{init} = \{s_i\}$ of N ANs as the initial starting set of SNs.
Output: Set $S = \{s_i\}$ of N ANs to operate as edge SNs and sets S_i with the ANs, assigned to SN $s_i, i = 1, \ldots, N$.

1. $S = S_{init}$
2. Determine S_i's: Assign every AN v_j in $V - S$ to its closest SN s_i in S
3. Set $TCC_{min} = TCC(S)$ and $LB_{min} = LB(S)$, as given by Equation (8) for S and S_i's
4. For $j = 1$ to M {
5. swap = FALSE
6. For $i = 1$ to N, $j \neq \text{ind}(s_i)$ {
7. Temporary swap v_j and s_i: $S_{tmp} = (S - \{s_i\}) \cup \{v_j\}$
8. Determine $S_{i,tmp}$'s: Assign every AN v_j in $V - S_{tmp}$ to its closest SN $s_{i,tmp}$ in S_{tmp}
9. if $(TCC(S_{tmp}) < TCC_{min})$ or $(TCC(S_{tmp}) == TCC_{min}$ and $LB(S_{tmp}) < LB_{min})$
10. set swap = TRUE; $i^* = i$; $TCC_{min} = TCC(S_{tmp})$; $LB_{min} = LB(S_{tmp})$
}
11. if swap == TRUE {
12. $S = (S - \{s_{i^*}\}) \cup \{v_j\}$
13. Determine S_i's: Assign every AN v_j in $V - S$ to its closest SN s_i in S
}
}
14. Exit;

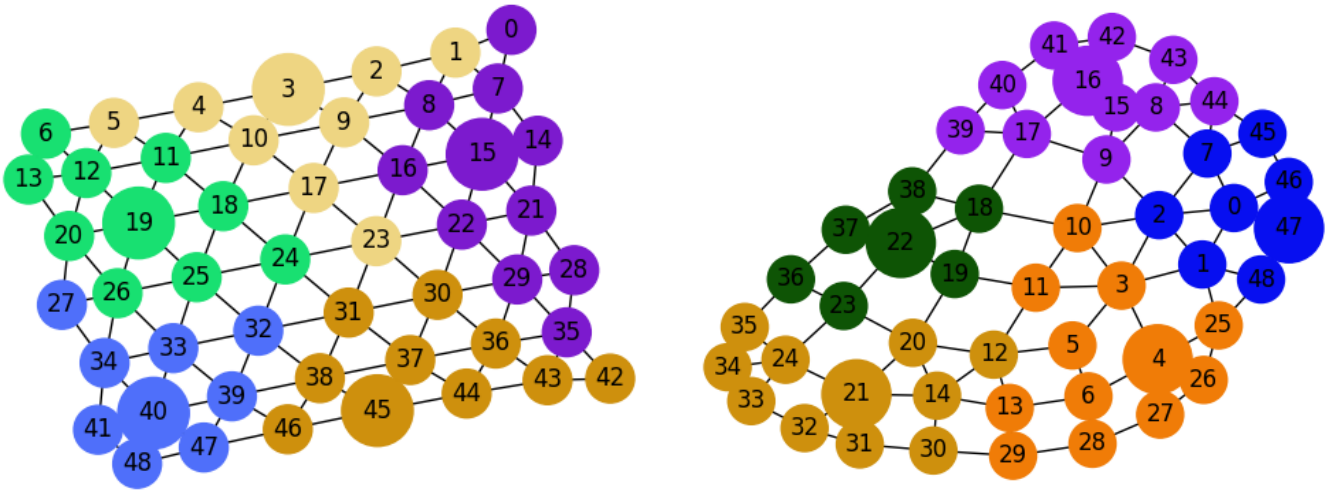
The time complexity of the Local Search implementation of Algorithm 5 is as follows. Devising an initial starting solution with Forward Greedy requires $O(M^2N^2)$. The loop 4-14 dominates lines 2 and 3 and is performed $M \times N$ times, while the lines 8 and 14 require $N(M - N)$ checks. Thus, the total time complexity of the procedure is $O(M(M - N)N^2)$. For low values of N, this is $O(M^2)$, for N around $M/2$, it reaches $O(M^4)$, and for high N values, it approaches $O(M^3)$.

5. Evaluation Results and Discussion

5.1. Experimental Environment

We evaluate the performance of the previously described approaches in 2 edge layer topologies: a 7×7 symmetrical, not wrapped, lattice (Figure 2a) of 49 ANs with higher connectivity among neighboring nodes (6 neighbors ANs per non-edge AN) and a random grid topology of 49 ANs with sparser connectivity (Figure 2b). In the same figure, the large circles indicate five operational SNs colocated with the corresponding ANs. The numbers in the circles denote the sequence number of AN. The topologies better represent the flat architecture of contemporary 5G and future generation mobile provider networks, rather than the past hierarchical structures.

Two experiments are conducted: (i) *uniform* user distribution across the network of ANs, in other words $w_i = 6$, $i = 1, \ldots, 49$, and (ii) *non-uniform* user distribution where different integer weights (w_i) varying from 3 to 9 are randomly assigned to 49 ANs of the networks. In the latter experiment, the results presented below are calculated by averaging the results of ten independent random weight allocations.



(**a**) (**b**)

Figure 2. Lattice and random grid graph topologies. (**a**) Edge Service Node Selection Load Balanced results (49 ANs, 5 SNs, uniform weights), (**b**) Local Search with Load Balance results (49 ANs, 5 SNs, non-uniform weights).

The schemes under evaluation are based on (a) node proximity only, where the decision which ANs are allocated to each SN is based solely on the communication cost between ANs and SNs, and (b) node proximity and load balance, a combined approach where both communication cost and even distribution of load across the SNs, are considered. Table 1 summarizes the schemes and their categorization.

The comparison is conducted in terms of communication cost, load balance, and their combination. Instead of using sample variance (Equation (8)) for the load balance metric, we present the load W_{max} allocated to the most loaded SN. In addition, all experiments were conducted on the same computer with 3.60 GHz Intel i7-4790 CPU (Dell PC, property of University of Piraeus, Piraeus, Greece), 16 GB RAM and 64-bit Windows 10 Pro and all algorithms were fully developed from scratch in Python 3.7 and compared in terms of computational times under the same conditions.

Table 1. Schemes under evaluation.

Scheme	Abbreviation	ANs to SNs Allocation Criterion	
		Node Proximity	Load Balance
Edge Service Node Selection (Algorithm 1) with Node Proximity AN Allocation	SNNP	✓	
Edge Service Node Selection (Algorithm 1) with Load Balance and Node Proximity AN Allocation (Algorithm 2)	SNLB	✓	✓
Forward Greedy (Algorithm 3 without line 5)	FG	✓	
Forward Greedy with Load Balance (Algorithm 3)	FGLB	✓	✓
Reverse Greedy (Algorithm 4 without line 4)	RG	✓	
Reverse Greedy with Load Balance (Algorithm 4)	RGLB	✓	✓
Local Search (Algorithm 5 without second OR clause in line 9)	LS	✓	
Local Search with Load Balance (Algorithm 5)	LSLB	✓	✓

5.2. Communication Cost of Node Proximity Schemes vs. Node Proximity with Load Balance Schemes

Figure 3 shows the communication cost perceived by ANs for both lattice and random grid topologies of 49 ANs and for uniform and non-uniform weights distributions. The measurement unit on the y-axis of these figures corresponds to the expected average communication cost delay per user between ANs and SNs of a network provider, which, in a real network this would translate into latency ranging from a few msecs up to a few tens of msecs.

A general observation of the node proximity schemes is that as the number of SNs increases, the communication cost decreases. As more and more SNs are instantiated, service is always offered closer to the ANs and thus delay is decreased. The best behavior is achieved by LS, which is always better than FG, since LS in our experiments use the solution outcome of FG as a starting point, and LS repeatedly attempts to improve it. The results of SNNP are very close to the LS and FG results. The behavior of SNNP is mainly justified by the fact that the Service Node Selection scheme of Algorithm 1 attempts to evenly distribute the SNs across ANs of the network, but closer to the most crowded ANs. Furthermore, a non-uniform distribution of load makes more difficult the task of evenly distributing the selection of the most appropriate SNs in SNNP; thus, the procedure works slightly better in uniform distribution of workload. RG on the other hand, behaves less effectively, especially in the lattice network for non-uniform and even worse in uniform load.

Likewise, the node proximity with load balance schemes present a similar behavior, with some differences. Again, LSLB and FGLB yield the same communication cost as LS and FG. This behavior is expected as the modified schemes favor more load balanced solutions that have the same communication cost. Thus, it is very rare to yield solutions with better load balance, but worse communication cost. RGLB, however, yields better communication cost solutions than RG, especially in scenarios with small number of SNs, high connectivity and uniform offered workload. This heuristic starts with a high number of SNs and continuously removes one SN, yielding a set of SNs with minimum communication cost and lower load balance.

It seems that the selection of a more balanced solution at some point yields a different search path and results in lower communication cost solutions. A different behavior is encountered when comparing SNNP and SNLB, which are differentiated only by the way ANs are assigned to SNs. For a small number of SNs the behavior is identical. With fewer SNs the choices are limited and it is more likely to assign the AN to its most close SN as SNNP does. However, as the number of SNs increases, the SNLB method to visit each SN in a round-robin manner and locate the next AN to assign for load balance, will eventually

result, after several cycles, in assigning remote ANs to some SNs. Thus, this leads to a deterioration in the goal of minimum communication cost.

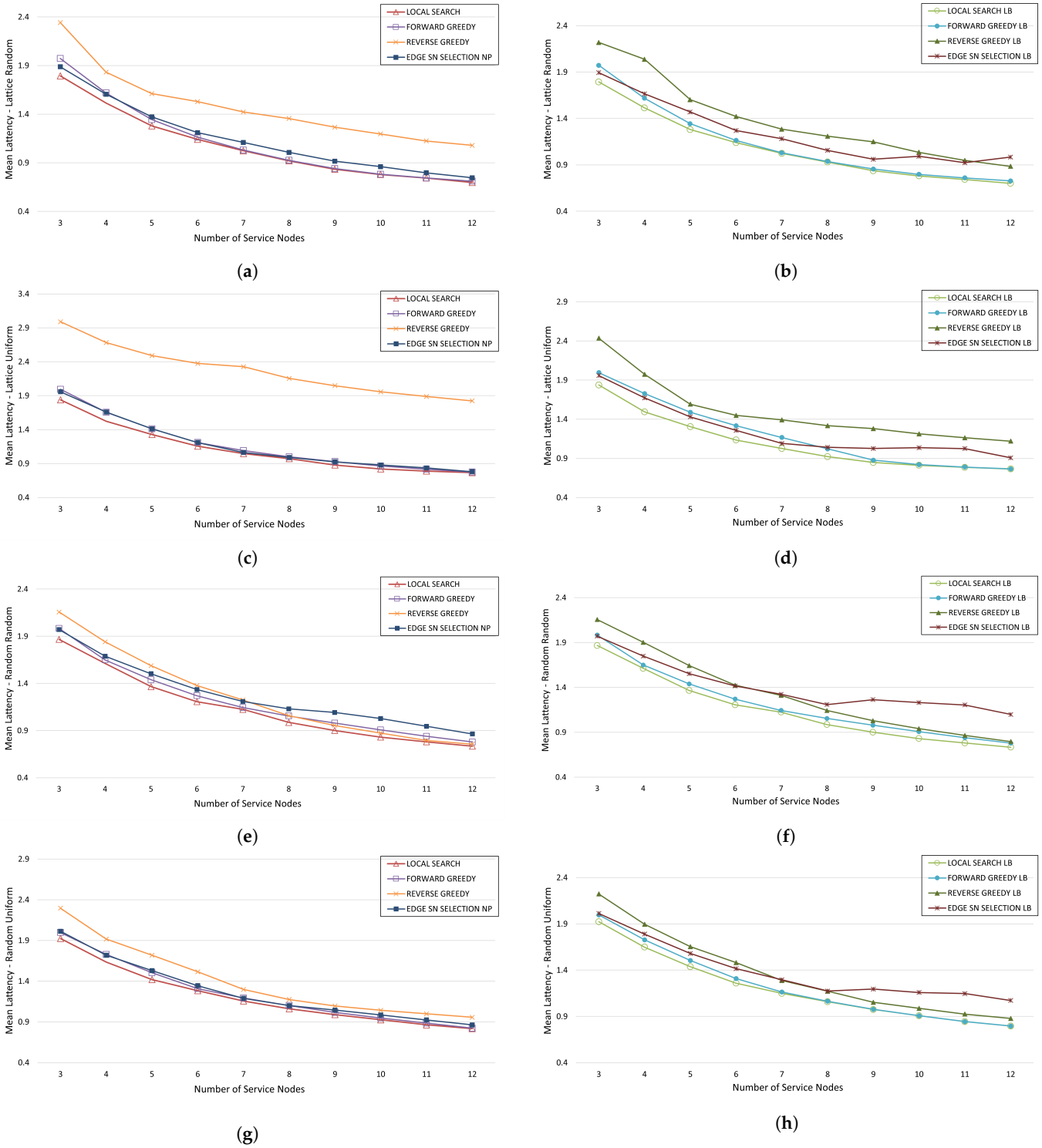


Figure 3. Average latency communication cost for lattice and random grid topologies. *Node Proximity schemes*: (**a**) Lattice topology and random weight distribution, (**c**) Lattice topology and uniform weight distribution, (**e**) Random grid topology and random weight distribution, (**g**) Random grid topology and uniform weight distribution, *Node Proximity with Load Balance schemes*: (**b**) Lattice topology and random weight distribution, (**d**) Lattice topology and uniform weight distribution, (**f**) Random grid topology and random weight distribution, (**h**) Random grid topology and uniform weight distribution.

5.3. Load Balancing of Node Proximity with Load Balance Schemes

Figure 4 shows the load balance behavior of the schemes for the same topologies and weights distributions. The metric shown is the load W_{max} allocated to the most loaded SN. Only node proximity with load balance schemes are presented, as they have shown superior performance than the simple node proximity schemes. It is necessary to note that allocations of ANs to SNs in the SNLB scheme are designed to favor balanced ANs allocations to close SNs, as described in Algorithm 2, in contrast to all other schemes that are adapted to favor more balanced solutions only if there is more than one choice when selecting a solution with minimum communication cost.

From the figures depicting the load of the most loaded SN, it can be observed that SNLB outperforms all other schemes and achieves the least maximum load in nearly all numbers of SNs at the lattice and random grid topologies of 49 ANs for both random and uniform load distributions (Figure 4a–d). In these figures, the mean load (dotted line) is a lower bound of the maximum load and SNLB results are closer to this bound. The SNLB scheme achieves superior workload balance through its round-robin design by prioritizing the even distribution of ANs' workload across nearby SNs. This approach minimizes the maximum load on any single SN, bringing it closer to the mean load value.

Another observation is that SNLB, FGLB, and LSLB behave much better in uniform loads. This is partly justified by the fact that in the examined system model high loads cannot be split and direct different parts of the loads to different SNs, which would produce more even allocations. Another justification for SNLB is that its round-robin behavior works better when each AN contributes equal loads and as a result an equal amount of load is assigned in each cycle. The worst behavior is presented by RGLB, especially in small and medium numbers of SNs, and this is more evident in the lattice topology rather than the random grid topology.

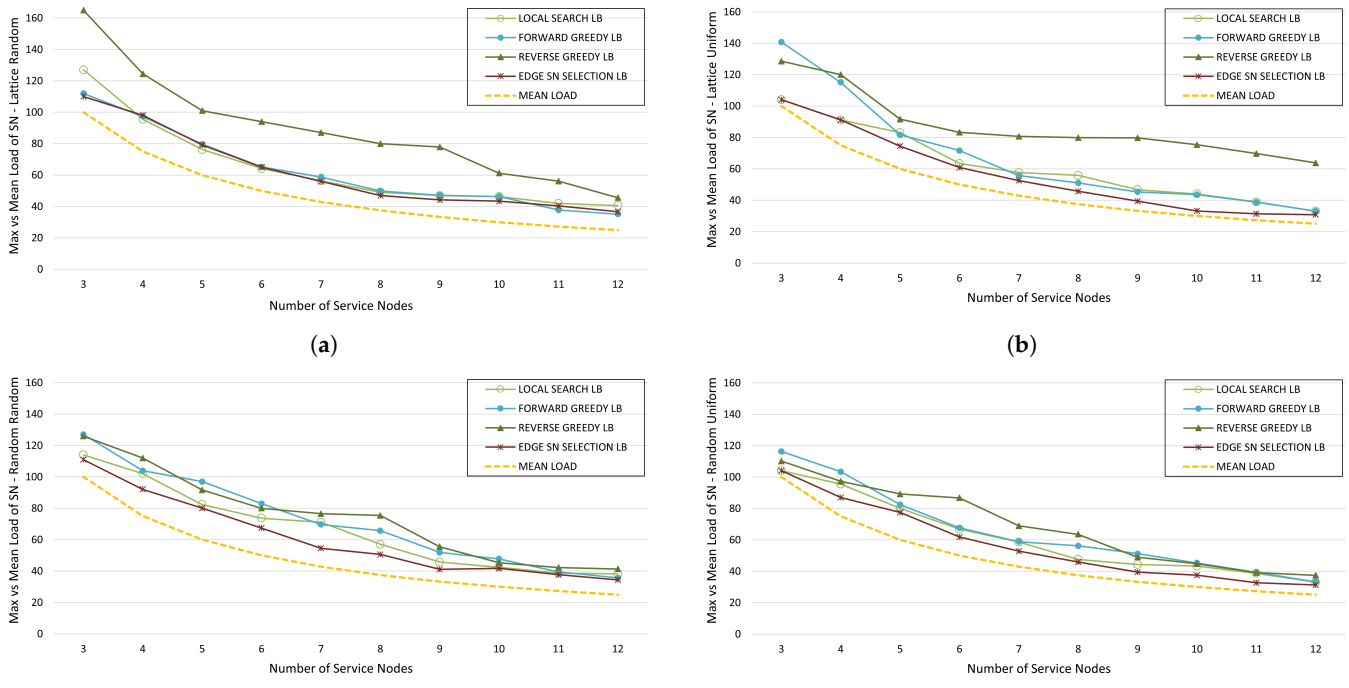


Figure 4. Load balance metric for lattice and random grid topologies with uniform and random load distributions. (**a**) Lattice topology and random weight distribution, (**b**) Lattice topology and uniform weight distribution, (**c**) Random grid topology and random weight distribution, (**d**) Random grid topology and uniform weight distribution.

5.4. Bi-Objective Function Results of Node Proximity with Load Balance Schemes

Figure 5 illustrates the behavior of node proximity with load balance schemes when both terms of communication cost and load balancing are equally weighted in the normalized bi-objective function (Equation (6)) ($\lambda_{cc} = \lambda_{lb} = 0.5$). FGLB, RGLB, and LSLB, by their inherent operation prioritize communication cost over load balance, while, SNLB focuses primarily on load balancing and secondarily on communication cost. A main observation is that SNLB outperforms the other schemes in most numbers of SNs when the load is uniform, while in non-uniform loads LSLB seems to behave slightly better than FGLB and SNLB in most cases. This can be justified as follows.

In uniform load scenarios, all ANs contribute equal load and SNLB, with its round-robin approach, manages to efficiently distribute ANs' load across SNs, while the other heuristics cannot benefit from communication cost by favoring most loaded ANs to serve as SNs. Thus, the gain of SNLB on the load balance term outperforms the lower gain of LSLB and FGLB on the communication cost term.

However, under conditions of non-uniform loads, where an AN's load is fully allocated to a specific SN, LSLB, as an enhancement of FGLB, exhibits slightly better performance by targeting the most heavily loaded ANs to serve as SNs, allowing it to better handle varying load distributions, where SNLB's even allocation becomes less efficient. In this situation, the benefits of LSLB from the communication cost term surpass the lower gains of SNLB from the load balance term.

Finally, RGLB is obviously worse, especially in the lattice topology for a small and medium number of SNs, following the same trend observed in the above load balance and average communication cost independent studies.

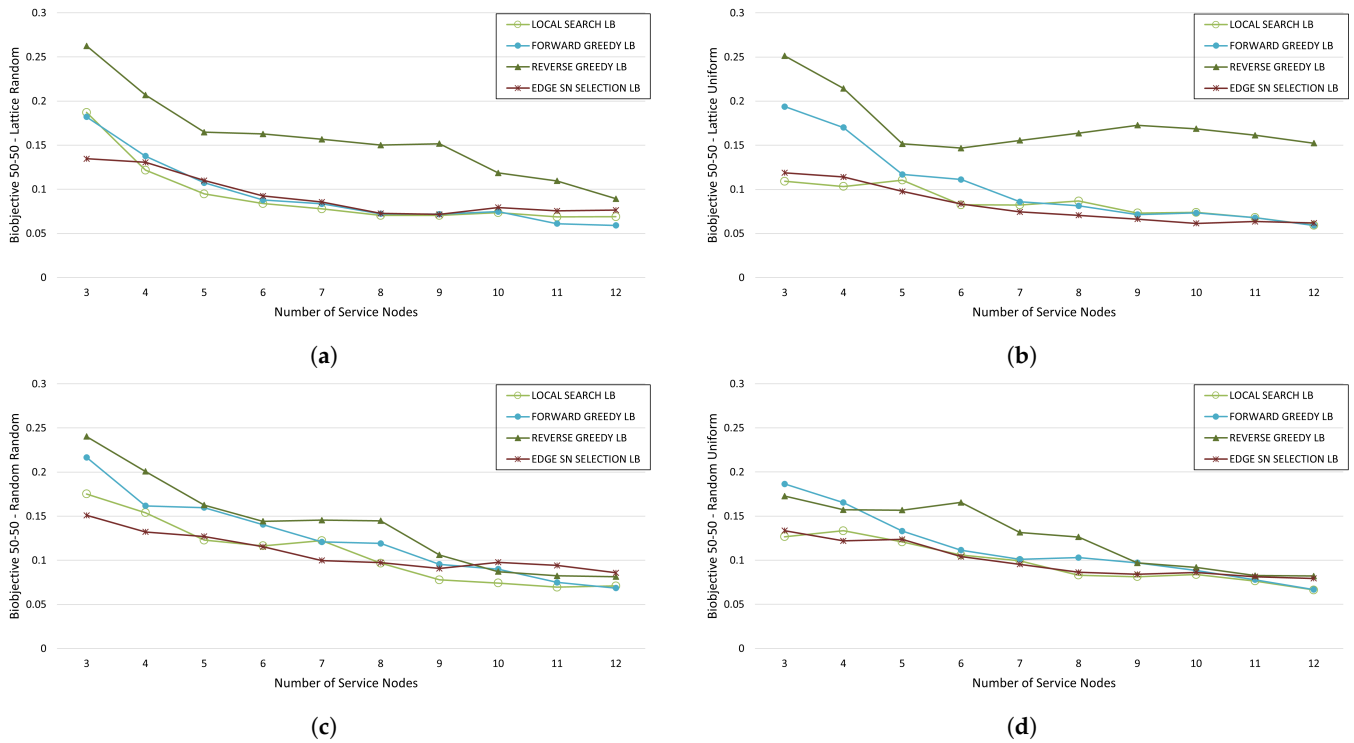


Figure 5. Equally weighted communication cost and load balance in bi-objective function of Equation (6). (**a**) Lattice topology and random weight distribution, (**b**) Lattice topology and uniform weight distribution, (**c**) Random grid topology and random weight distribution, (**d**) Random grid topology and uniform weight distribution.

5.5. Computational Times

Finally, the computational performance of all schemes is depicted in Figure 6. To avoid noisy results, due to other workload running at the same time in the host computer, we

show the averages of the outcomes of 500 different runs. We present the results for the lattice topology with random distribution of load only, as all the other results are quite similar and follow the same trend. Figure 6 depicts the total time to compute up to the number of SNs shown in the figure. As we can see, when the SNs are few, the SN set is determined quite fast with FG and FGLB, while LS and LSLB are slower because these include the respective FG and FGLB times. Furthermore, it takes much longer for RG and RGLB to determine the required SNs when the number of SNs is small. This is justified by the way RG searches for SNs, according to which it starts with all SNs as operational and then tries to remove SNs one by one. Thus, when the number of final operational SNs is small, the scheme needs to remove more SNs. On the other hand, SNNP and SNLB are extremely fast compared to all the other schemes and the most important observation is that running times exhibit great stability despite the number of SNs determined.

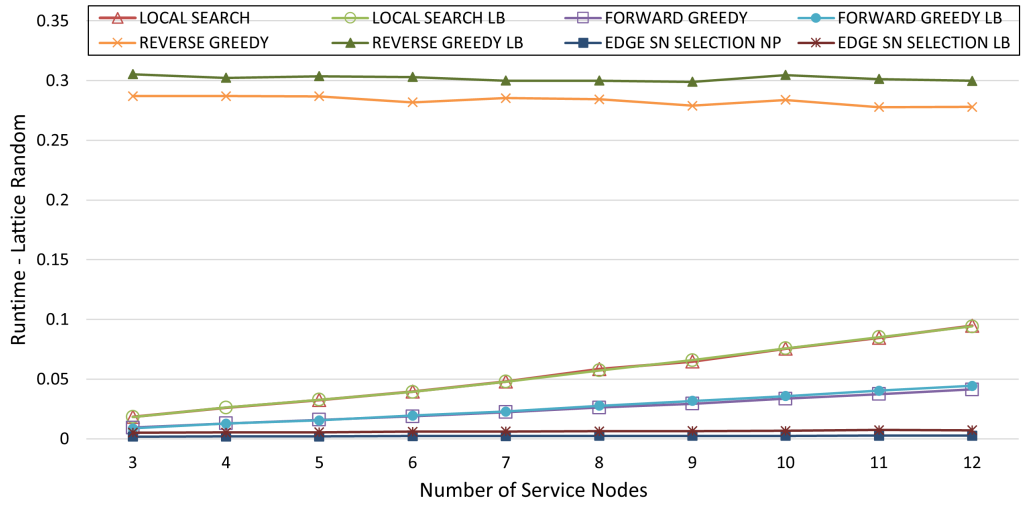


Figure 6. Computation times for lattice topology of 49 ANs and random load distributions.

6. Conclusions

In this paper, we introduced a novel approach to tackling the dual optimization challenge of minimizing communication costs while simultaneously balancing workloads within edge network topologies. We formulated two distinct optimization problems and combined them into a single bi-objective function. To solve these, we introduced heuristic schemes for Edge Service Node Selection and Access Node Allocation to Service Nodes. Extensive simulations were conducted across both lattice and random grid network topologies with uniform and non-uniform workload distributions, demonstrating the effectiveness of our proposed schemes in optimizing both objectives of communication costs and load balancing.

SNLB consistently outperforms other schemes in terms of load balancing, achieving the lowest maximum load across various topologies and load distributions, particularly under uniform conditions. Its round-robin design, enhanced with proximity-based decisions, ensures an even distribution of workload, bringing the maximum load closer to the mean. Additionally, SNNP, while slightly less efficient in load balancing, shows competitive performance in minimizing communication costs, particularly in small-scale scenarios of service nodes. Overall, our schemes are robust and efficient solutions, adaptable to specific network requirements and objectives, with their scalability and efficiency further demonstrated by superior execution times across all experimental scenarios. Future research can build on these findings and enhance these schemes with intelligent decision making to accommodate time- and spatial-varying user load conditions.

Author Contributions: Conceptualization, A.R.; methodology, E.O. and A.R.; software, E.O.; validation, E.O. and A.R.; formal analysis, E.O. and A.R.; investigation, E.O.; writing—original draft

preparation, E.O. and A.R.; writing—review and editing, E.O. and A.R.; supervision, A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article material, further inquiries can be directed to the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

5G	5th Generation
AN	Access Node
ANA	Access Node Allocation
AR	Augmented Reality
BTS	Base Transceiver Station
CAPABLE	Cost Aware cloudlet PlAcement in moBiLe Edge computing
CAPEX	Capital Expenditures
ELBS	Energy-aware Load Balancing and Scheduling
FG	Forward Greedy
FGLB	Forward Greedy with Load Balance
IoT	Internet of Things
IoV	Internet of Vehicles
IT	Information Technology
LAB	LoAd Balancing
LS	Local Search
LSLB	Local Search with Load Balance
MEC	Mobile Edge Computing
MHP2P	Mobile Hybrid hierarchical Peer-to-Peer
OPEX	Operating Expenditures
QoE	Quality of Experience
QoS	Quality of Service
RG	Reverse Greedy
RGLB	Reverse Greedy with Load Balance
SN	Service Node
SNLB	edge Service Node selection with Load Balance and node proximity AN allocation
SNNP	edge Service Node selection with Node Proximity AN allocation
SNS	Service Node Selection
WMAN	Wireless Metropolitan Area Network

References

1. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
2. Wu, Q.; Wang, W.; Fan, P.; Fan, Q.; Wang, J.; Letaief, K.B. URLLC-Aware Resource Allocation for Heterogeneous Vehicular Edge Computing. *IEEE Trans. Veh. Technol.* **2024**, *73*, 11789–11805. [CrossRef]
3. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
4. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile edge computing: A key technology towards 5G. *ETSI White Paper* **2015**, *11*, 1–16.
5. Reznik, A.; Arora, R.; Cannon, M.; Cominardi, L.; Featherstone, W.; Frazao, R.; Giust, F.; Kekki, S.; Li, A.; Sabella, D.; et al. Developing Software for Multi-Access Edge Computing. *ETSI White Paper* **2017**, *20*, 1–16.
6. Bonomi, F.; Milito, R.A.; Natarajan, P.; Zhu, J. Fog computing: A platform for Internet of Things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Cham, Switzerland, 2014; pp. 169–186.
7. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the Internet of Things. In Proceedings of the ACM SIGCOMM Workshop on Mobile Cloud Computing, Helsinki, Finland, 13–17 August 2012; pp. 13–16.
8. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J.P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330. [CrossRef]
9. Cao, K.; Liu, Y.; Meng, G.; Sun, Q. An Overview on Edge Computing Research. *IEEE Access* **2020**, *8*, 85714–85728. [CrossRef]

10. Ahmed, A.; Ahmed, E. A survey on mobile edge computing. In Proceedings of the 2016 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 7–8 January 2016; pp. 1–8.
11. Klas, G.I. Fog Computing and Mobile Edge Cloud Gain Momentum Open Fog Consortium, ETSI MEC and Cloudlets. 2015. Available online: https://yucianga.info/wp-content/uploads/2015/11/15_11_22-_Fog_computing_and_mobile_edge_cloud_gain_momentum_Open_Fog_Consortium-ETSI_MEC-Cloudlets_v1_1.pdf (accessed on 19 August 2020).
12. 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]
13. Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; Zhou, T. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* **2019**, *107*, 1537–1562. [CrossRef]
14. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2018**, *6*, 6900–6919. [CrossRef]
15. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Internet Things J.* **2018**, *5*, 450–465. [CrossRef]
16. Filali, A.; Abouaomar, A.; Cherkaoui, S.; Kobbane, A.; Guizani, M. Multi-Access Edge Computing: A Survey. *IEEE Access* **2020**, *8*, 197017–197046. [CrossRef]
17. Wang, S.; Xu, J.; Zhang, N.; Liu, Y. A Survey on Service Migration in Mobile Edge Computing. *IEEE Access* **2018**, *6*, 23511–23528. [CrossRef]
18. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
19. Kong, L.; Tan, J.; Huang, J.; Chen, G.; Wang, S.; Jin, X.; Zeng, P.; Khan, M.; Das, S.K. Edge-computing-driven Internet of Things: A Survey. *ACM Comput. Surv.* **2022**, *55*, 174. [CrossRef]
20. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
21. Jia, M.; Cao, J.; Liang, W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Trans. Cloud Comput.* **2017**, *5*, 725–737. [CrossRef]
22. Wang, S.; Zhao, Y.; Xu, J.; Yuan, J.; Hsu, C.-H. Edge server placement in mobile edge computing. *J. Parallel Distrib. Comput.* **2019**, *127*, 160–168. [CrossRef]
23. Guo, Y.; Wang, S.; Zhou, A.; Xu, J.; Yuan, J.; Hsu, C.-H. User allocation-aware edge cloud placement in mobile edge computing. *Softw. Pract. Exp.* **2020**, *50*, 489–502. [CrossRef]
24. Kasi, S.K.; Kasi, M.K.; Ali, K.; Raza, M.; Afzal, H.; Lasebae, A.; Naeem, B.; Islam, S.; Rodrigues, J.J. Heuristic edge server placement in industrial internet of things and cellular networks. *IEEE Internet Things J.* **2021**, *8*, 10308–10317. [CrossRef]
25. Asghari, A.; Sohrabi, M.K. Multiobjective Edge Server Placement in Mobile-Edge Computing Using a Combination of Multiagent Deep Q-Network and Coral Reefs Optimization. *IEEE Internet Things J.* **2022** *9*, 17503–17512. [CrossRef]
26. Jia, M.; Liang, W.; Xu, Z.; Huang, M.; Ma, Y. QoS-Aware Cloudlet Load Balancing in Wireless Metropolitan Area Networks. *IEEE Trans. Cloud Comput.* **2020**, *8*, 623–634. [CrossRef]
27. Jia, M.; Liang, W.; Xu, Z.; Huang, M. Cloudlet load balancing in wireless metropolitan area networks. In Proceedings of the IEEE INFOCOM 2016, San Francisco, CA, USA, 10–14 April 2016.
28. Wan, J.; Chen, B.; Wang, S.; Xia, M.; Li, D.; Liu, C. Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4548–4556. [CrossRef]
29. Meng, J.; Shi, W.; Tan, H.; Li, X. Cloudlet Placement and Minimum-Delay Routing in Cloudlet Computing. In Proceedings of the 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, China, 10–11 August 2017; pp. 297–304.
30. Xu, Z.; Liang, W.; Xu, W.; Jia, M.; Guo, S. Efficient algorithms for capacitated cloudlet placements. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 2866–2880. [CrossRef]
31. Li, Y.; Wang, S. An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 66–73.
32. Ceselli, A.; Premoli, M.; Secci, S. Mobile Edge Cloud Network Design Optimization. *IEEE/ACM Trans. Netw.* **2017**, *25*, 1818–1831. [CrossRef]
33. Fan, Q.; Ansari, N. On cost aware cloudlet placement for mobile edge computing. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 926–937. [CrossRef]
34. Sun, X.; Ansari, N. Green cloudlet network: A sustainable platform for mobile cloud computing. *IEEE Trans. Cloud Comput.* **2018**, *8*, 180–192. [CrossRef]
35. Lähderanta, T.; Leppänen, T.; Ruha, L.; Lovén, L.; Harjula, E.; Ylianttila, M.; Riekki, J.; Sillanpää, M.J. Edge computing server placement with capacitated location allocation. *J. Parallel Distrib. Comput.* **2021**, *153*, 130–149. [CrossRef]
36. Fan, Q.; Ansari, N. Towards workload balancing in fog computing empowered IoT. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 253–262. [CrossRef]
37. Duan, Z.; Tian, C.; Zhang, N.; Zhou, M.; Yu, B.; Wang, X.; Wu, Y. A novel load balancing scheme for mobile edge computing. *J. Syst. Softw.* **2022**, *186*, 111195. [CrossRef]

38. Liu, H.; Wang, S.; Huang, H.; Ye, Q. On the Placement of Edge Servers in Mobile Edge Computing. In Proceedings of the 2023 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 20–22 February 2023; pp. 496–500.
39. Oikonomou, E.; Rouskas, A. Selection of Service Nodes in Edge Computing Environments. In Proceedings of the 2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Paris, France, 14–16 December 2020; pp. 1–6.
40. Oikonomou, E.; Rouskas, A. Optimizing load balancing and minimizing communication latency in edge networks. In Proceedings of the 2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON), Porto, Portugal, 25–27 June 2024; pp. 820–825.
41. Li, S. On Uniform Capacitated k-Median Beyond the Natural LP Relaxation. *ACM Trans. Algorithms* **2017**, *13*, 1–18. [CrossRef]
42. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
43. Han, J.; Kamber, M.; Pei, J. Data Mining: Concepts and Techniques. In *The Morgan Kaufmann Series in Data Management Systems*; Morgan Kaufmann: Burlington, MA, USA, 2012.
44. Charikar, M.; Guha, S.; Tardos, É.; Shmoys, D.B. A constant-factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci.* **2002**, *65*, 129–149. [CrossRef]
45. Chrobak, M.; Kenyon, C.; Young, N. The reverse greedy algorithm for the metric k-median problem. *Inf. Process. Lett.* **2006**, *97*, 68–72. [CrossRef]
46. Arya, V.; Garg, N.; Khandekar, R.; Munagala, K.; Pandit, V. Local search heuristic for k-median and facility location problems. In Proceedings of the 33rd ACM Symposium on Theory of Computing, Hersonissos, Greece, 6–8 July 2001; pp. 21–29.