Edge computing server placement with capacitated location allocation

Tero Lähderanta a,∗, Teemu Leppänen b,1, Leena Ruha a,d,1, Lauri Lovén b,1, Erkki Harjula c, Mika Ylianttila c, Jukka Riekki b, Lauri Lovén b,1, Erkki Harjula c, Mika Ylianttila c, Jukka Riekki b

a Research Unit of Mathematical Sciences, University of Oulu, Finland
b Center for Ubiquitous Computing, University of Oulu, Finland
c Centre for Wireless Communications, University of Oulu, Finland
d Natural Resources Institute Finland, Oulu, Finland

A R T I C LE I N F O

Article history:
Received 16 April 2020
Received in revised form 11 February 2021
Accepted 17 March 2021
Available online 6 April 2021

Keywords:
Multi-access edge computing
Facility location
Clustering
k-medoid

A B S T R A C T

The deployment of edge computing infrastructure requires a careful placement of the edge servers, with an aim to improve application latencies and reduce data transfer load in opportunistic Internet of Things systems. In the edge server placement, it is important to consider computing capacity, available deployment budget, and hardware requirements for the edge servers and the underlying backbone network topology. In this paper, we thoroughly survey the existing literature in edge server placement, identify gaps and present an extensive set of parameters to be considered. We then develop a novel algorithm, called PACK, for server placement as a capacitated location-allocation problem. PACK minimizes the distances between servers and their associated access points, while taking into account capacity constraints for load balancing and enabling workload sharing between servers. Moreover, PACK considers practical issues such as prioritized locations and reliability. We evaluate the algorithm in two distinct scenarios: one with high capacity servers for edge computing in general, and one with low capacity servers for Fog computing. Evaluations are performed with a data set collected in a real-world network, consisting of both dense and sparse deployments of access points across a city area. The resulting algorithm and related tools are publicly available as open source software.

© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Edge computing has emerged as a scalable and flexible solution to address the challenges introduced by cloud-centric Internet of Things (IoT) systems [59]. Cloud is physically and logically far away from the things, and requires massive upstream data uploads, consuming core network resources and introducing latencies. Moreover, system monitoring and control from a distance is challenging in the swiftly opportunistic IoT environment.

Edge computing introduces new computational layers atop the network infrastructure, between the cloud and the IoT devices, to where virtualized application resources can be moved. In close proximity to the devices, edge platforms have the capabilities to address the dynamics of the environment and serve local computation- and data-intensive applications with low latency and high bandwidth. Further, data processing at the edge layer increases data privacy and reduces load in the networks towards the cloud.

Such edge platforms have already been presented [6,15,59]. (i) Cloudlets [46], i.e. small-scale data centers, host mobile applications as virtual machines at the edge layer, (ii) Multi-access Edge Computing (MEC) [45], previously known as Mobile Edge Computing, reuses network infrastructure components, such as cellular network base stations and Wi-Fi access points (AP), for data storage and hosting mobile applications, and (iii) Fog computing [7] harnesses local networked devices, e.g. Wi-Fi APs and gateways but also laptops and set-top boxes, to form virtualized hierarchical computing platforms.

However, resource provisioning in the edge introduces pragmatic complexities. For example, network operators need to consider the placement of physical edge platform components such as servers. Often, constrained deployment budgets result in a limited number of servers with varying capacity and hardware. One option is to share server and network resources across the deployment, but a trade-off is that latencies are introduced and network load increased. Furthermore, some locations, for example commercial centers or research facilities, may have deployment priorities over other locations. Finally, the deployment area may cover not only densely populated areas such as city centers, but also sparsely populated suburban areas. Without systematic approaches, the only way to solve these pragmatic issues is to rely on the domain expertise of the network operator.

∗Correspondence to: University of Oulu, P.O. Box 8000, FI-90014, Finland.
E-mail address: tero.lahderanta@oulu.fi (T. Lahderanta).
1 These authors contributed equally.

https://doi.org/10.1016/j.jpdc.2021.03.007
0743-7315/© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
Having decided on an initial server placement scheme, the next challenge is the online allocation and balancing of edge application workloads across the deployment. These two connected problems have been typically handled in steps, where workloads are allocated and balanced atop the initial server placement scheme, e.g. [25,55]. The parameters for server placement are often derived from the observed online workload. Some works also focus solely on optimizing the online workload, e.g. [38,43,47,56,61]. In such cases, the design of the underlying system is a significant factor, since the initially selected deployment scheme affects system performance in response to the online workload and latencies [51]. The initial deployment decisions are thus pivotal in managing mobile applications across the multi-tenant edge platform to provide a high Quality of Service (QoS) for the edge platform and Quality of Experience (QoE) for mobile users. An edge server placement scheme, which aims for maximal on-line QoS and QoE, involves two interrelated problems: first, the placement of edge servers, and second, the allocation of computing capacity to those servers to suit their expected workload (Fig. 1). In this paper, we focus on solving these two problems optimally. First, we provide a survey of existing solutions for physical edge server placement and allocation, and identify gaps in those solutions. Second, based on the gaps identified, we develop a novel placement and allocation algorithm, called 

PACK, which takes into account an extensive set of parameters, including latencies, workload balancing and sharing, server co-location with APs, and prioritized locations. The resulting deployments are capable of handling the maximum workloads observed in the system while reaching both minimum latencies between the APs and the edge servers as well as a balanced load between the edge servers. The deployments thus ensure fair QoS across the deployment, and distribute application workloads such that the server capacities are respected, with minimal over- or under-provision. To increase the flexibility of the deployment, the workloads of APs can be shared between servers, addressing the problem of overloaded low capacity servers. Finally, the algorithm allows preferring certain server locations over others, and providing reserve computational capacity as a reliability measure for edge application execution.

These considerations lead to an NP-hard complex optimization problem which seeks for the best compromise between latency and workload balance. We consider this placement problem as an instance of the generic location–allocation framework [18]. We adapt the framework for both MEC and Fog edge platforms, with high and low capacity servers respectively, and evaluate the resulting placement schemes with a set of performance statistics. The schemes are developed with a real-world public Wi-Fi network dataset [29] that consists of user equipment (UE) connections in both urban and suburban areas.

The contributions of this article are:

- A detailed survey of existing physical edge server placement solutions. An extensive set of required parameters are collected and resulting gaps identified in the optimization of server deployments.
- A location–allocation algorithm to provide optimal edge server placement with clustering of APs, based on the total system workload and number of servers, that minimizes the geospatial distances in the system, while considering server capacity constraints, workload balancing and sharing, reserve capacity and location preference. These additional constraints demonstrate real-world challenges and options faced in planning of edge deployments, which we consider as parts of critical edge infrastructure.
- A study of different edge deployment scenarios, i.e. MEC and Fog, in relation to the above set of parameters with a set of proximity statistics, to evaluate QoS between the presented algorithm and solutions in previous work.
- Development and evaluation of the placement algorithm, based on a massive-scale real-world data set (∼257M data points) in a city-wide public Wi-Fi network, covering an area of a whole city with both dense and sparse AP deployments.
- An open source software package,² implemented in R, of the algorithm.

The rest of the paper is organized as follows. In Section 2 we introduce and discuss the state-of-the-art in edge server placement, as well as the identified gaps in the research. In Section 3 we present our algorithm, which we evaluate in Section 4. We utilize a real-world data set in different scenarios and compare the results to the schemes based on previous work. Lastly, in Section 5 we discuss the results and, in Section 6, we present the conclusions.

2. Background

Previous studies address edge server placement from four perspectives: (1) minimizing latencies, which are approximated using a number of different distance measures; (2) minimizing the deployment costs of the servers while limiting maximal latency; (3) optimizing the trade-off between latency and deployment costs; or (4) maximizing the user connections, i.e. coverage, within the clusters. The studies consider different sets of functional parameters for placement, such as the number of servers, individual server capacity, geo-locations of servers, and location priorities. Also, non-functional properties of QoS and QoE such as reliability have been used as parameters in the studied placement algorithms.

The studies applied varying approaches for placement: k-means [34], k-means with mixed-integer quadratic programming [21], density-based clustering [23], graph theory as in a minimum dominating set problem [60], hierarchical tree-like structures [24,49], multi-objective constraint optimization [5,14,53], mixed integer linear programming [9,17], integer linear programming with heuristic solutions [31,54,57], non-linear

2 https://github.com/terolahderanta/rpack.
mixed integer programming \cite{41}, clustering combined with optimization as a facility location problem \cite{40}, and with heuristic algorithms \cite{37,58}.

In comparison, PACK places a fixed number of servers while minimizing the latencies between users and edge servers, balancing the system workload and satisfying the lower and upper limits of server capacity. PACK can be seen as a variant of k-means type clustering problem with capacity constraints, solved here with a block coordinate descent algorithm with integer programming steps.

### 2.1. Survey of edge server placement approaches

Next, we present a detailed survey of the existing solutions for physical edge server placement and analyze the set of parameters utilized in the algorithms. Further, we discuss the properties of our novel placement algorithm, PACK, presented in Section 3, addressing the identified gaps. The overall properties of the solutions are summarized in Table 1.

| Article | Workload | Capacity constraints | Distance Measure | Role of distance | Number of servers | Server association |
|---------|----------|----------------------|------------------|------------------|-------------------|-------------------|
| Bhatta et al. \cite{5} | workload | upper | netw. topology | thres. & min. | minimized | hard |
| Bouet et al. \cite{9} | phone calls | upper | geospatial | not used | on-demand | fixed |
| Chen et al. \cite{11} | simulated | upper | netw. topology | thresh. & min. | fixed | request |
| Cui et al. \cite{13} | total no. of requests | not used | geospatial | not directly used | on-demand | request |
| da Silva et al. \cite{14} | time series | threshold | geospatial | minimized | on-demand | fractional |
| Fan et al. \cite{17} | requests | upper | netw. topology | thresholded | fixed | request |
| Gedeon et al. \cite{19} | simulated | upper | geospatial | minimized | on-demand | hard |
| Guan et al. \cite{20} | peak | threshold | geospatial | minimized | fixed | hard |
| Guo et al. \cite{21} | total requests | balanced | netw. topology | minimized | fixed | hard |
| Jia et al. \cite{23} | request rate | balanced | netw. topology | thresholded | fixed | hard |
| Jiao et al. \cite{24} | workload | not used | geospatial | minimized | fixed | hard |
| Kang et al. \cite{25} | total requests | not used | geospatial | minimized | fixed | request |
| Leyva-Pupo et al. \cite{31} | simulated | upper | geospatial | thres. and min. | on-demand | duplicate |
| Li et al. \cite{33} | simulated | upper | netw. topology | minimized | fixed | request |
| Li et al. \cite{32} | total conn. time | upper | geospatial | thresholded | on-demand | hard |
| Liu et al. \cite{34} | not used | not used | geospatial | minimized | fixed | hard |
| Ma et al. \cite{39} | request rate | upper & balanced | netw. topology | minimized | fixed | request |
| Meng et al. \cite{40} | total requests | upper | netw. topology | thresholded | minimized | hard |
| Mohan et al. \cite{41} | average requests | not used | netw. topology | minimized | fixed | hard |
| Mondal et al. \cite{41} | request rate | upper | netw. topology | thresholded | minimized | hard |
| Siniky et al. \cite{40} | total requests | not used | netw. topology | minimized | fixed | hard |
| Wang et al. \cite{53} | total requests | balanced | geospatial | minimized | fixed | hard |
| Xu et al. \cite{54} | total no. of requests | upper | netw. topology | minimized | fixed | request |
| Yao et al. \cite{57} | total requests | upper | netw. topology | minimized | on-demand | request |
| Yin et al. \cite{58} | peak | upper | geospatial | thresholded | on-demand | hard |
| Zeng et al. \cite{60} | workload | upper | netw. topology | thresholded | minimized | hard |
| This article | peak | upper & lower | geospatial | minimized | fixed | hard or frac. |

2.1.2. Computing capacity

Computing capacity refers to the computational and data resources of individual edge servers, available for the workload accumulating from the APs. Deciding upon capacities requires considering a trade-off between balanced workload and low latency. Indeed, minimizing the latency between APs and edge servers, the resulting deployment may have high workloads in busy areas and low workloads in quiet areas, leading to either unused capacity or edge servers with varying capacities. Conversely, balanced workloads, distributed evenly among edge servers, may lead to long distances between AP’s and edge servers, manifesting as increased latencies.

In related works, where a predefined number of servers were placed, typically either capacity limits, workload balance, or both were neglected. In some works, however, the servers had varying capacities. In more detail, most studies addressed computing capacity by prohibiting over-provisioning with a hard upper constraint on server workload. Alternatively, workload was balanced by distributing workload across the deployment \cite{21,23,53}, server capacity was scaled up on-demand regardless of the resulting cluster size \cite{34,40,49,57,60}, or neighboring server capacity was utilized as a reserve or duplicated capacity \cite{11,31}. As a catch-all solution, if server capacity could not match its workload, the remaining load was discarded or offloaded to a cloud \cite{9,14,23,39,41}, where resources were assumed to be unlimited.

A further alternative, cloudlet technology, allows scaling up server capacity as a backup measure to adapting to changing online workloads \cite{14,23,41}. However, an initial cloudlet placement is still required, where additional capacity increases deployment and maintenance costs of the operator. A related solution is to support already deployed servers by adding new ones atop the existing deployment \cite{35}.
Server placement schemes have mostly considered densely populated central areas (e.g. [13,17,19,23,40,53]), but in some works also suburban areas were included (e.g. [17,23,31,40,41,58]). The trade-off between capacity and latency issue is exacerbated if the infrastructure has a high contrast between busy central areas and less busy suburban areas.

PACK supports both minimizing the latencies as well as balancing the workloads, ensuring sufficient server capacities, by applying both upper and lower capacity limits. Using both limits allows strong control over the latency–workload trade-off. For example, by prioritizing balance, one can set the capacity limits close to each other around the average workload. Alternatively, with sufficiently wide capacity limits, the lower limit can represent the capacity of one small server, and the upper limit the capacity of a large server.

Further, a cloudlet can be built based on several low capacity servers. Thus, a wide interval allows flexibility with deployment areas with high variability in workloads, and more granularity in individual server capacity.

2.1.3. Performance metrics

The performance of an edge server or the deployment is typically measured though a set of QoS and QoE parameters. ETSI standardization considers multiple metrics for demonstrating the performance deployments with both functional and non-functional sets of Key Performance Indicators (KPIs) [16], where the use of metrics depends on the use case. Functional metrics include latency, throughput, packet delivery and energy efficiency. Non-functional metrics include service availability, reliability and fault tolerance.

Most of the studies measured performance by the latency between servers and users and/or APs, using different approaches. First, if the number of deployed servers was fixed, the average or total latency between APs and servers was used [11,13,21,23,24,33,34,37,49,53,54,57]. Additionally, the standard deviation of the server-wise workloads was used to evaluate load balancing [21,53]. Second, if the aim was to minimize the number of servers or their energy consumption, the minimization was constrained by latency thresholds [32,40,41,60].

Other performance measures in the studies considered both QoE and QoS aspects, including the percentage of satisfied user requests and the total utilization of the server capacity [40], the percentage of users within the expected distance [58], the score and the cost of the deployment [31], intra-cluster traffic [40], energy consumption and average resource utilization [32], and the ratio of how much user demand can be allocated to cloudlets [14,19].

Surveyed works evaluate the effect of various parameters to the QoS. First, the effect of the number of servers to latency [21,23,24,34,49,53,54], to workload balance [21,53] and to hosted workload [14] have been evaluated. Second, the effect of the capacity constraints to intra-cluster traffic, and of temporal changes to workload balance [9], as well as the effect of the number of APs to average access delay [54] and to the energy consumption and average resource utilization [32] have been evaluated. Third, the link between other parameters and deployment budget has been evaluated, including the effect of distance thresholds [20,32,60] or capacity constraints [9,19,60] to the number of servers. Further, Yin et al. [58] modeled the cost of the deployment as a function of the percentage of people within a given distance from the server, while Gedeon et al. [19] considered the fixed costs of deploying an edge server, as well as the variable costs of using them based on the workload. Finally, Fan et al. evaluated deployment cost and latency as a function of the parameter that controls the trade-off between budget and proximity [17].

In summary, the surveyed articles measure deployment performance typically either (1) as the average QoS or (2) through setting a threshold for the worst tolerable QoS. However, our approach is to use multiple simultaneous performance measures, providing a clearer view of the resulting QoE and QoS. In more detail, we measure both the average and the worst-case latencies between the APs and servers, providing insights into the QoE across the whole coverage of the deployment. Further, we measure the resulting workloads on the edge servers, providing insights into the QoS of the resulting deployment.

2.1.4. Latency

The communication latency between the APs and the edge servers is one of the main KPIs. Latency in an edge infrastructure, typically within a metropolitan area network, results from the architecture of the core network, i.e. its topology, and link capacities between the nodes. However, in all surveyed works the core network topology or the link capacities were not available. As a result, latency was approximated using computationally light proxies, such as topological or geophysical distances, depending on what kind of information was available.

With simulated topology [11,17,21,23,24,37,39–41,49,54,57,60], latency was either approximated by hop counts, or no specific latency measure was used at all. Alternatively, latency was approximated by the geospatial distances between the APs and the edge servers [9,14,20,31–34,53,58]. However, this requires that the coordinate space is consistent with respect to distance ranking [58]. Most studies used Euclidean distances, while some [33,34] relied on squared Euclidean distances. Further, Wang et al. [53] scaled the Euclidean distances with respect to the density of the access points, while Leyva-Pupo et al. [31] et al. discretized the distances on multiple levels.

PACK is agnostic to the distance metric that represents latency, as discussed in detail in Section 3.2. In more detail, PACK accepts any pairwise distance between the possible server locations and the APs.

Like with all surveyed studies, the underlying physical network topology is not available in the data we use for evaluation [29]. Hence, evaluating PACK, we approximate latency with the geospatial distances between APs and servers. To provide fair QoS across the deployment, we use squared Euclidean distances, as they have a tendency to produce spherical clusters with centralized heads [42]. The use of squared distances results in a star-like topology with spatially centralized servers, contributing towards better worst-case proximity across the deployment, that is, lower latencies, regardless of the topological and physical positions of APs in the network.

2.1.5. Number of edge servers

The number of edge servers in the deployment is a trade-off between cost (e.g. operator budget and resulting available server capacity) and performance in relation to the user requirements.

The surveyed studies relied on three main approaches to optimize the trade-off. First, based on a budget, a fixed number of servers were placed in a way that the latencies between the APs and the servers were minimized [11,13,19,21,23,24,33,34,37,39,49,53,54]. Second, the highest tolerated latency was defined and the number of servers was minimized with such latency constraint for each AP [40,41,57,60]. Third approach was to optimize the trade-off between the cost and the QoE, as number of satisfied users [14,58], or the QoS of the resulting deployment [5,11,17,20,31]. Other approaches included determining the number of servers as the result of minimizing energy consumption [32] or maximizing connections within server regions [9].

In addition to the above trade-off, the deployment area needs to be considered. With city-wide real-world data sets, as in Section 4.1, some deployments cover not only densely populated central areas, but also sparsely populated suburban areas, both
with different requirements for server capacity. In such a deployment, minimizing the number of servers while satisfying latency constraints is no longer feasible, as in sparse areas the distances between servers and APs vary significantly, while remaining short in the denser areas. Therefore, such a placement scheme could lead to a large number of servers with a small number of connected APs.

Acknowledging that the network operator is often operating under a strict budget, PACK follows the majority of the surveyed studies and assumes a fixed number of servers. We evaluate PACK with the above-mentioned heterogeneous city-wide data set, covering both dense and sparse areas. As discussed by Wang et al. [51], the deployment scheme of the servers must be considered. Indeed, when communication latency can be relatively high, sparse servers in a flat architecture are preferred. However, if latency must be small and workloads are large, a hierarchical deployment architecture can increase edge system performance. Considering both viewpoints, we evaluate PACK in two different scenarios, named MEC and Fog, the former with a small number of high-capacity servers and the latter with a large number of low-capacity servers.

2.1.6. Co-location of servers and APs

Deployment and maintenance costs of network operators are lower, and no new infrastructure is needed, when edge servers are co-located with APs. Most surveyed studies followed this constraint. As exceptions, co-location was not considered by some studies [9,34,58], while Bouet et al. considered spatial server areas, without exact server locations [9]. Finally, Liu et al. [34] allowed the servers to be placed anywhere, and Yin et al. [58] considered new, possibly unforeseen, server locations.

PACK follows the majority of the studies, co-locating edge servers with APs.

2.1.7. Workload allocation

The expected system workload, originating from user requests, propagates through the network from APs to edge servers. Most of the surveyed studies allocated the workload passing through individual APs to exactly one server, with some exceptions. Bouet et al. left some APs unallocated, without an edge server, as the overall workload of the servers exceeded their total capacity [9]. Jiao et al. [24] placed a fixed number of servers in a tree-structure, leaving some APs unassociated. With cloudlets, the workload of user requests was delivered to available cloudlets across the deployment [11,14,19,25,33,37,39,54,57]. As a last resort, excessive workload was offloaded to the cloud. Da Silva et al. consider two types of workload, strict which can only be processed in a server and flexible that can be either hosted in the server or in the cloud [4].

Fan et al. [17], on the other hand, shared AP workload among several servers. In more detail, when the total workload on a server exceeds its capacity, workload must be shared between servers if offloading to cloud is not feasible. Particularly, this is beneficial with widely available low-capacity servers, e.g., laptops, as exemplified by Fog computing, where small-scale workload from local user requests can be handled in the premises. Shared workload can also relieve the trade-off between latency and workload balance, and improve the scalability of the placement algorithm. On a downside, whenever server capacity is available in premises, sharing complicates the management of the deployment and increases core network load. However, the studies did not consider the effects of both strict allocation and sharing of workload simultaneously.

Adopting clustering terminology, we refer to workload sharing as fractional membership and non-shared workload as hard membership. PACK supports selecting the membership type depending on the scenario. Thus, both flat and hierarchical architectures [31], e.g., MEC and Fog, hosting applications with different latency and workload requirements, are considered in placement.

2.1.8. Location preference

Based on the business model or domain expertise of the network operator, city centers and other densely populated areas are often initially considered as edge server locations. Moreover, domain expertise and practical considerations, e.g., to guarantee a low latency to the edge server, may dictate a set of preferred locations, e.g. an airport, a commercial center, or a research facility, even when the location could not otherwise be justified. Moreover, in a smaller scale, site-specific requirements, e.g. privacy concerns in a Fog deployment, may dictate placement of servers within a facility. Such fixed placement candidates can result in sub-optimal performance of the deployment, particularly with a limited budget. Further, as a practical handicap, network topology may constrain the locations in which the servers can be placed. In the surveyed studies, Leyva-Pupo et al. addressed location preference with site-specific latency requirements for the APs, and by reserving duplicate capacity for servers at prioritized APs [31]; Yao et al. on the other hand, placed a set of servers according to combined user mobility patterns [57].

PACK considers preferred edge server locations, based e.g. on domain expertise or practical issues, in both placement and allocation. K-means type clustering approaches are weight sensitive, that is, points with large weights attract a cluster center more strongly than light weighted points [1]. In our approach, the total weight of an AP is the sum of its workload and the location preference value for that AP, giving the preferred locations reduced latency in the placement. Thus, location preference is supported in the algorithmic level by adding a predefined constant to the workloads of the most critical APs. This approach further allows the use of several preference classes, or even a preference variable, where the size of the constant depends on the value of the variable.

2.1.9. Reliability

A system that is reliable and robust performs as designed and required for extended periods of time, and can resist and recover from failures. Shi et al. [48] discuss reliability challenges for edge infrastructure, where component failures can cause network and service interruptions. Further, user mobility and intermittent connectivity makes it challenging to provide a service and maintain QoE. Therefore, system-level management and recovery of failures is a crucial concern. Indeed, ETSI considers reliability as one of the non-functional KPIs for edge services [16].

In the surveyed studies, Leyva-Pupo et al. [31] address reliability in workload allocation by providing backup capacity in servers for every AP. This way, if a server fails, a replacement is readily available. A drawback is that the deployment costs increase and possibly unused capacity will be reserved.

Further, Cui et al. [13] address the concern of runtime failures in edge server networks by optimizing the placement of edge servers based on a trade-off between user coverage and network robustness. They form a placement strategy with weight parameters for both user coverage and robustness that represent the preference between those two parameters.

To address reliability, PACK allows the workload of APs to be replicated and assigned to any number (typically two) of servers instead of just one. Minding budget constraints, the decision on workload replication can be made for each AP individually, selecting only those APs that are deemed as critical network infrastructure by a domain expert.

2.1.10. Data sets

Either simulated or real-world data sets were used for the development and evaluation of the placement algorithms. All the presented data sets were ca. five years old, even those used in the more recent studies. To the best of our knowledge, data sets of large-scale edge computing infrastructures do not exist yet.
Multiple studies used simulated data sets \cite{11,17,20,24,31,33,37,41,57,60}, with the number of APs varying from tens to hundreds. The simulation was sometimes based on additional data such as city population data \cite{5,23,49,54} or the locations of cell towers, public Wi-Fi APs and street lamps, as well as user mobility traces \cite{19}.

A number of studies based their evaluations on real-world Wi-Fi or mobile network data sets. The geo-referenced phone call records in Milan, Italy, from November 2013 to January 2014 \cite{3} was the most popular \cite{9,14,34,40}, while a Shanghai Telecom data set with 4.6 million call records and 7.5 million movement traces of 10 thousand mobile users in 3000 base stations, over a period of six months in 2014 \cite{52}, came a close second \cite{21,32,53}. Yin et al. obtained their data from the globally-distributed PlanetLab nodes as well as measurement nodes deployed in China \cite{58}. Guan et al. utilized the traces of 320 taxi cabs in Rome, Italy, during one month in 2014 \cite{20}.

We evaluate PACK with a real-world data set collected across a city-wide open Wi-Fi network deployment in Oulu, Finland. The data set consists of 450 Wi-Fi APs, deployed into both central and suburban areas, and all of their connected users during the year 2014 (≈257M data points). The data set is described in detail in Section 4.1.

### 2.1.11. Algorithmic scalability

Algorithmic scalability refers to how the running time of the algorithm depends on the data. In general, the better results the algorithm provides, the worse it scales. Some studies presented heuristic algorithms \cite{11,37,54,58}, that have good scalability but with no guarantees on result optimality. Most of the studies relied on formal optimization methods which, while more complex, guarantee that the result is at least locally optimal. An exception, Ma et al. propose particle swarm optimization to address the scalability of an initially heuristic algorithm \cite{37}.

For formal optimization methods, the fewer constraints in the optimization, the less control the user has but the better the algorithm scales. The simplest optimization methods had no capacity constraints \cite{25,34,49}, being scalable but with the least control. The studies that included optimization constraints improved scalability with an iterative approach where location and allocation steps alternated in an iterative optimization algorithm. Leyva-Pupo et al. \cite{31} used for large-scale data sets variants of hybrid simulated annealing and evolutionary algorithms to alternatively optimize placement and allocation with capacity constraints.

Some studies first partitioned the workload of APs into clusters, without applying capacity constraints, after which the servers were placed at each cluster separately \cite{40,58}. Alternatively, servers were first placed, without workload balancing, and then APs allocated to servers with the aim to balance the workload \cite{21,23}. Further, Guo et al. \cite{21} used quadratic optimization for the balanced allocation step, thus reducing computation time but consuming memory as the whole data is allocated at once. Jia et al. relied on a hybrid model, using a fast but heuristic approach for the balanced allocation step \cite{23}. Also, Guan et al. utilize a two-step method where the area was first partitioned based on minimizing the service migration costs, and then the location of servers and the allocation of APs was optimized separately \cite{20}.

Bouet et al. \cite{9} set a dense grid in the designated area, and then merged the grid cells by their workloads to obtain the spatial extents of the servers. The computational time thus depended on the number of cells and not on the number of APs. Thus, the method scales well in relation to the number of APs, but not with respect to the spatial extent.

PACK relies on formal optimization, and we improve the scalability of our approach by a block-coordinate descent algorithm with integer programming steps. The chance of local optima are reduced by repeating the optimization with a large number of initial values. We also allow fractional membership, improving scalability. The scalability of our approach depends on the number of APs but not on the spatial extent of the deployment coverage. This is important as we focus on large-scale infrastructure with both busy city centers and quiet suburban areas.

#### 2.1.12. Source code

None of the surveyed studies has published the source code for their methods. PACK source code is published as an open source project, called pack, in GitHub \cite{30}. The algorithm is implemented with R and the package contains all the described functionality with an easy-to-use API.

### 2.2. Summary

In summary, the PACK algorithm advances the previous studies as follows:

- Placing a fixed number of edge servers, PACK does two things simultaneously. Latencies between APs and servers are minimized, and the workload of the edge servers is balanced, satisfying both lower and upper capacity constraints of the servers.
- PACK allows both wide and tight capacity ranges and sharing the workload of APs, leading to flexible and scalable deployment in different scenarios, depending on the requirements and server capacity. This is particularly beneficial in Fog computing with low-capacity servers.
- PACK allows designating APs as preferred, i.e., to be located close to an edge server. The preference of APs was largely omitted in the related work, except with site-specific latency requirements by Leyva-Pupo et al. \cite{31}.
- PACK addresses reliability concerns in the deployment by providing reserve capacity for APs that are considered a part of critical network infrastructure. In the related work, double capacity for the whole deployment was considered by Leyva-Pupo et al. \cite{31}, but otherwise omitted.
- PACK is evaluated with latency statistics on both QoS and QoE, using sample quantiles, and including both average and worst case latencies.
- The development and evaluation of PACK relies on a large-scale real-world data set, which includes both dense central and sparse suburban areas with wide range of workloads.
- PACK is evaluated on scenarios with both a low number of high-capacity edge servers (“MEC”) and a high number of low-capacity servers (“Fog”).

In Table 2 we compare how our approach compares to others related to the gaps identified.

### 3. PACK algorithm for edge server placement

In this section, we present in detail the PACK algorithm for physical edge server placement. The PACK algorithm optimizes the placement of a fixed number of edge servers, minimizing the distances between servers and APs while balancing system workload, and satisfying server capacity constraints with both lower and upper limits. The algorithm maintains fair QoS, regardless of the AP deployment density, in a dispersed city-wide data set. PACK considers QoS with a pairwise distance measure between APs and servers, such as latency or hop distance, if available, or geospatial distance. Further, PACK provides a number of parameters which model workload sharing, workload replication, and allow preferring certain locations over others. Finally, we consider the evaluation of PACK results by way of QoE and QoS metrics.
3.1. The PACK-algorithm

Given a data set of n APs, let \(x_i\) be the coordinates for AP \(i\). Further, AP \(i\) has workload \(w_i\), which can be determined by, for example, the average or total amount of connected users per AP (see Section 2.1.1).

Furthermore, the total weight value \(a_i\) for AP \(i\) is defined as:

\[
a_i = w_i + \gamma_i,
\]

where \(\gamma_i\) is the location preference of the AP \(i\), set for example by a domain expert (see Section 2.1.8). The default value for \(\gamma_i\) is 0, that is, no preference is set. For the preferred APs, we set the value of \(\gamma_i > 0\). The value of \(\gamma_i\) can be considered as artificial workload added to AP \(i\). The higher the value of \(\gamma_i\), the bigger the total weight of AP \(i\) is in the optimization.

We place a fixed number \(k\) of edge servers, aiming to minimize weighted distances to their associated APs and satisfying the capacity constraints. Such a placement scheme can be considered as a capacitated location allocation problem \([10,12,18]\). Let us denote by \(c_j, j = 1, 2, \ldots, k\), the coordinates for the \(k\) edge servers, each of which is assumed to have similar capacity for computational load.

Denote by \(y_{ij}\) the membership of the AP \(i\) to the edge server \(j\). In other words, \(y_{ij}\) is an indicator variable of the allocation of the workload, where \(y_{ij} = 1\) if the workload in AP \(i\) is allocated to the server \(j\) and \(y_{ij} = 0\) otherwise. We address the reliability of the placement by assigning selected APs to more than one (say, two) servers (see Section 2.1.9; cf. the Q-coverage problem \([26]\)). Lastly, \(d(i, j)\) denotes the distance between AP \(i\) and server \(j\). The algorithm allows the use of any pair-wise distance metric \(d\), as discussed in Section 3.2.

Our goal is to minimize the sum of weighted distances between the edge servers and the APs they cater for, while taking into consideration the workload of each AP and the capacity constraints of each server. Hence, we minimize the objective function

\[
\min_{c_j,y_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{k} a_i d(i,j) y_{ij}
\]

with the following four constraints:

1. \(c_j \in \{x_1, x_2, \ldots, x_n\}\) \(\forall j\).

   Each edge server location \(c_j\) is assumed to be the same as one of the AP locations \(x_i\) (see Section 2.1.6).

2. \((a) y_{ij} \in [0, 1] \) or \((b) y_{ij} \in [0, 1]\) \(\forall i, j\).

   Either hard \((a)\) or fractional \((b)\) membership constraints are used (see Section 2.1.7). In the case of hard membership, the workload of an AP \(i\) is either completely allocated to edge server \(j\) \((y_{ij} = 1)\), or not at all \((y_{ij} = 0)\). In the case of fractional membership, \(y_{ij}\) represents the fraction of workload of AP \(i\) assigned to server \(j\) \([8]\).

3. \(\sum_{j=1}^{k} y_{ij} = q_i \forall i\).

   All the workload of an AP is assigned to \(q_i\) servers, where \(q_i > 1\) for APs deemed critical by a domain expert, and \(q_i = 1\) for others (see Section 2.1.9). In case of hard membership, \(q_i\) must be an integer.

4. \(L \leq \sum_{j=1}^{k} w_j y_{ij} \leq U \forall i\).

   Capacity for each edge server is constrained between \(L\) and \(U\) (see Section 2.1.2).

For solving the optimization problem, we use a block-coordinate descent algorithm \([50]\). In Algorithm 1, we iterate two main steps: (i) the allocation of the APs to the given locations of servers (line 4) and (ii) re-locating each server given the APs allocated (line 5). The iteration is continued until the locations of the edge servers do not change.
Algorithm 1 PACK-algorithm
Input: $x_i, w_i, k, N$
Output: Edge server locations $c_j^{*}$ and allocations $y_{ij}^{*}$

1: for $l = 1$ to $N$
2: Initialize $c_j, j = 1, 2, ..., k$ using k-means++
3: while $c_j$ changes do
4: Allocation-step: minimize (1) with respect to $y_{ij}$
5: Location-step: minimize (1) with respect to $c_j$
6: $S$ ← the value of the objective function
7: end while
8: if $S < S_{\text{min}}$ or $l = 1$ then
9: $S_{\text{min}}$ ← $S$
10: $c_j^{*}$ ← $c_j$
11: $y_{ij}^{*}$ ← $y_{ij}$
12: end if
13: end for
14: return $c_j^{*}, y_{ij}^{*}$

As the result of such an iteration is not guaranteed to be the global minimum, $N$ initial values for the edge server locations are used in Algorithm 1. A set of reasonable initial values is obtained using the k-means++ algorithm [2]. The k-means++ algorithm spreads the initial server locations, which improves both the speed and the accuracy of the k-means method.

**Allocation-step.** In the allocation-step, we minimize the objective function (1) with respect to $y_{ij}$. Only constraints 2, 3 and 4 are used, since edge server locations $c_j$ are assumed to be fixed. For the hard membership constraint, this step is an NP-hard integer programming task [44]. However, in the case of the fractional membership, this is a linear programming task that can be solved in polynomial time [28].

**Location-step.** Location-step relocates the edge servers based on the allocation of the APs. In other words, the objective function (1) is minimized with respect to $c_j$ with only constraint 1, while keeping the allocations $y_{ij}$ fixed. This step is identical to calculating for each server the medoid of the APs assigned to it, where medoid is defined as the point from which the sum distances to all the points in the cluster is minimized [27].

We implemented the block-coordinate descent algorithm as an open source R package called rpack [30]. The allocation step is run on Gurobi [22], a fast optimizer package with R bindings. If Gurobi is not available, the allocation-step can be run on the lpSolve package for R [4].

3.2. Distance

In the related work, typical choices for the distance measure include the Euclidean and squared Euclidean distances,

$$d_1(i, j) = \|x_i - c_j\|,$$
$$d_2(i, j) = \|x_i - c_j\|^2,$$

respectively.

In the case of Euclidean distances the problem can be considered as the capacitated p-median problem [18] and the squared Euclidean distances as a capacitated k-means clustering, where the cluster centers are constrained to the data points. Such a discrete variant of k-means method is generally referred to as a k-medoids method [27].

By applying the Euclidean distances in (1), the average distances form the servers are minimized. However, the minimization of the squared Euclidean distances imposes a stronger penalty from large deviations, thus there is a tendency for more spherical-like clusters with centralized cluster heads [42]. Such a scheme results in a star-like deployment topology with spatially centralized edge servers. This contributes towards better latency, i.e. fair QoS for both dense and sparse AP deployments, being particularly beneficial for otherwise remote APs. Thus, we prefer the squared Euclidean distances and utilize them in our evaluations.

The PACK algorithm is agnostic towards the choice of the distance metric. By default, our open source rpack R package allows the use of both Euclidean and squared Euclidean distances. However, providing rpack with an $n \times n$ matrix of all pairwise distances between APs, any distance metric can be applied. This makes it possible to use metrics derived from the network topology, such as hop counts or latency metrics, if available. Even more generally, one can optimize any quantitative KPI that can be defined as a sum of pairwise functions of the APs and the servers they are assigned to. An example are objective and subjective edge service-dependent metrics as presented in [16], including the type of service, either real-time and non-real-time, or responsiveness as an QoE metric, which set their requirements for the placement candidates.

3.3. Evaluation metrics

We evaluate PACK with a set of both functional and non-functional KPIs, related to the QoS and the workload accumulated on the servers. Specifically, we focus on the distribution of latency between APs and edge servers, and the distribution of workload on the edge servers.

We measure latency with the selected distance metric, i.e., the distances between APs and servers. However, the same QoS metric can be interpreted as a QoE metric from a user point-of-view, i.e., as latency measured by how close the users are to the servers, or by way of the APs they are connected to. For the sake of interpretability, we apply the Euclidean distance as QoS metric as opposed to the squared Euclidean distance used in the placement.

The average AP distance weighted by the workload is obtained as

$$\text{Mean} = \frac{1}{W} \sum_{i=1}^{n} \sum_{j=1}^{k} w_i \|x_i - c_j^{*}\| y_{ij}^{*},$$

where $c_j^{*}$ are the optimized server locations, $y_{ij}^{*}$ are the optimized memberships and $W = \sum_{i=1}^{n} w_i$ is the total workload of the area.

Further, we explore the distance distribution thoroughly with the sample quantiles $q_{\alpha}$, which measure the maximal distance between AP and its edge server for $\alpha$ proportion of the workload. Hence, by selecting a proportion $\alpha$ close to 1, the worst case QoS can be evaluated. For example, $q_{0.95}$ is the distance from 95% of the workload to the server. Further, $\alpha = 0.5$, $q_{0.5}$ corresponds to the median QoS.

Finally, we measure the workload balance of the placement candidates with the standard deviation of the edge server workloads, as well as the minimum and maximum workload values.

4. Evaluation

To evaluate the PACK algorithm, we simulate edge server placement with a real-world Wi-Fi network data set in two edge computing scenarios: one with a small number of high-capacity edge servers (the MEC scenario), and the other with a large number of low-capacity edge servers (the Fog scenario). For the analysis, these two scenarios provide different real-world based viewpoints and validation criteria to evaluate both the flexibility of the PACK algorithm and the effects of the extensive set of parameters available in PACK. The aim of the evaluation is to provide insights into the resulting edge computing architecture, in terms of number of servers and their capacity ranges, and system
design factors such as scalability and costs in deployment and management, as discussed for example in [51]. Further, the evaluation considers a set of additional real-world based constraints on server capacity and workload sharing, location priorities, and reliability by way of workload replication.

Both the MEC and Fog scenarios present a number of setups which highlight different aspects of placement. In all setups, PACK uses 100 initial placement candidates (random sets of server locations), selected with the k-means++ method. Final server placement and the allocation of APs to edge servers are based on clustering with the smallest value of the objective function (Eq. (1)), as described in the PACK algorithm (Algorithm 1).

4.1. Data

The evaluations for the MEC and Fog computing scenarios are conducted as simulations, based on a real-world data set, collected from panOULU public network Wi-Fi APs in the city of Oulu between the years 2007–2015 [29]. As presented in Section 2.1, related work also present data sets on the same time period.

In the evaluation, we concentrate on data collected during the full year 2014, as it was observed to have a maximum number of yearly Wi-Fi connections (in total 257,552,871) on 450 active APs. The AP deployment, as shown in Fig. 2(a), consists of both a densely covered city center and sparsely covered suburban areas in the City of Oulu, Finland.

Each edge server can handle limited computational load. We assume that each connected user brings a load of one on the edge server. Aggregating a unitary user workload over the hundreds of millions of connections in the recorded data offers a simple yet effective method for estimating the relative workloads between APs, used in the offline edge server placement.

Fig. 2(c) illustrates the number of simultaneously connected users in an APs of a local polytechnic as a time series in different scales, i.e., yearly, monthly, and weekly. For example, on the weekly connections panel (bottom), the difference between working days and weekend can be clearly seen. On the yearly panel (top), the concentration of the connections at the start of the semester, after the summer months, is clearly visible. The highest number of simultaneous connections in 2014 for that particular AP reaches ca. 100.

![PanOULU AP deployment](image1)

![Maximum number of connected users on each AP](image2)

![Time series of the workload of one AP](image3)

Fig. 2. Location and workload of each AP.
We define the workload of an AP as the highest number of simultaneous connections to that AP in 2014. This way we emphasize the maximum workload during peak hours. As shown in Fig. 2(b), the distribution of such maximum workloads is heavily skewed, with a large number APs with low workloads and small number APs with exceedingly high workloads.

PACK admits any pairwise distance metric. However, as with all related work, the topology of the underlying network was not available in the data set. In a heterogeneous data set such as the one used here, consisting of central and remote areas, the geographical distances between hops vary significantly. Such heterogeneity is typical, for example, in data sets featuring metropolitan areas. Typically, a single hop corresponds to a longer geospatial distance in a suburban area than in a dense city center. This makes geospatial distance a biased estimate of the underlying hop distance. However, the center areas with heavy workloads tend to overshadow the remote areas with less workload, resulting in unequal QoS between them. To favor the suburban areas, we exaggerate the distances there by using the squared Euclidean distance as an estimator of the underlying topology.

### 4.2. Edge server placement for MEC

The MEC scenario is based on Multi-access Edge Computing, with medium scale data centers that are commonly co-located with mobile network base stations [15,45], and more recently APs. Powerful edge servers, as part of the standardized MEC infrastructure, control the computational and communication resources of the connected network infrastructure components, including base stations, APs, switches and routers, etc., each with a set of connected UEs. The resulting edge computing infrastructure is therefore based on small set of geographically scattered clusters, with powerful edge servers as cluster heads, each with a large number of connected APs.

The MEC scenario assumes a budget of 20 edge servers for simulation, based on approximation with domain expertise. We center the capacity limits around the even workload (426.5) with the median latency (α% = 50%) as the lowest of all setups. Further, as both the upper and the lower capacity constraints are applied in the setup, M1 has excellent workload balance. However, the upper quantile latencies are clearly higher than in the other scenarios, resulting in the highest mean latency. Due to the uneven distribution between workload and AP densities, this combination of good workload balance and high worst case latency is further evident in Fig. 3, where the high worst-case latencies manifest as widely differing spatial areas of the edge server clusters: the wider the coverage of an edge server, the worse the latencies to that server from the APs at the borders. Nevertheless, the result shows that PACK does not under- or over-provision server capacity, regardless of the resulting geospatial coverage.

Conversely, setup M2, with no capacity constraints, has excellent latency with spatially equally sized clusters, but highly varying server workloads with a very high standard deviation. In such a setup, edge servers need to be heterogeneous, with varying capacity, to handle the resulting very low and very high workloads.

Setup M3, without a lower capacity constraint, presents a middle ground between M1 and M3. Setup M3 has four servers with workload under 327, the lower limit of scenario M1. As a result, M3 has lower latency than M1 for the large quantiles. On the other hand, due to the upper capacity constraint, M3 has better workload balance than M2. Accordingly, the resulting clusters vary in spatial size less than those of M1, but more than in M2 (Fig. 3).

Setup M4 introduces fractional membership together with both upper and lower capacity constraints. Fractional membership allows the workload of APs to be shared among a number of edge servers. The resulting placement candidate is nearly identical to M1 in terms of workload balance and latency, with just a slight improvement in the best case QoS, that is, the 25% quantile of relative latency. This results from the increased leeway in the

| Setup | k  | frac | cap   | Relative latency | Workload |
|-------|----|------|-------|------------------|----------|
|       | k  | frac | cap   | Mean             | cap      |
|       |    |      |       | 25%             | 50%      | 75%     | 95%     |
|       |    |      |       | S.D.            | Min      | Max     |
| M1    | 20 | no   | up/lw | 0.59            | 0.69     | 0.28    | 0.81    | 2.11    | 90      | 327     | 526     |
| M2    | 20 | no   | no    | 0.40            | 0.02     | 0.29    | 0.48    | 1.33    | 336     | 55      | 1013    |
| M3    | 20 | no   | up    | 0.55            | 0.10     | 0.38    | 0.71    | 1.61    | 162     | 57      | 526     |
| M4    | 20 | yes  | up/lw | 0.59            | 0.04     | 0.28    | 0.70    | 2.11    | 89      | 327     | 526     |
| M5    | 20 | no   | up/lw | 0.59            | 0.10     | 0.28    | 0.73    | 2.11    | 89      | 327     | 526     |
| M6    | 20 | no   | up/lw | 0.63            | 0.10     | 0.30    | 0.85    | 2.19    | 95.2    | 327     | 526     |

M6 Both lower and upper capacity constraints are present with enhanced reliability for five APs.

Except with setup M4, we default to hard membership, assuming the edge servers will not share the AP workloads. In M5 and M6, we assume that the APs are selected based on domain expertise and may be, for example, part of critical infrastructure. Location preference is achieved by setting the parameter γ = 100 for the selected APs. Reliability, by way of replicated workload, is achieved by assigning five APs to exactly two servers, i.e. q = 2 for the 5 critical APs.

Quantitative evaluation metrics (see Section 3.3) are collected in Table 3. Further, the resulting optimal server placement candidates for each setup in the MEC scenario are shown in Fig. 3 and Fig. 4, highlighting the trade-off between latency and workload balance.

Table 3

Evaluation results of the six MEC setups, comparing relative latency between edge servers and APs, and workload balance on the edge servers. k = number of servers, frac = fractional membership.

cap = capacity constraints (up = upper constraint, low = lower constraint).

Mean = the average AP distance, weighted by the workload. α% = the maximum distance between AP and edge server for α% of the workload. Workload balance is measured with the standard deviation of the server workloads (S.D.) and the minimum (Min) and the maximum (Max) obtained server workload. See Section 3.3 for further information. The green background indicates the best performance in terms of the different quality measures.
T. Lähderanta, T. Leppänen, L. Ruha et al. Journal of Parallel and Distributed Computing 153 (2021) 130–149

Fig. 3. MEC server placement scenarios. Crosses denote the server locations. WI: workloads of the servers. The color shades visualize the server regions. Quantitative measures of the scenarios can be seen in Table 3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4

| Setup | Relative latency |
|-------|------------------|
|       | AP id            |
|       | 800   588 712 974 693 |
| M1    | 40.4 162.6 93.4 122.0 29.2 |
| M5    | 40.4 82.9 76.3 37.7 29.2 |

Further, Fig. 4 shows some spatial overlap of the clusters due to the division of workload on some APs.

Setup M5 has five pre-selected, preferred locations for edge servers. As shown in Table 4, M5 ends up with three server locations closer to the preferred locations than in scenario M1. As a result, the preferred server locations enjoy a much lower latency, at the cost of a slightly worse best-case QoS, that is, the 25% quantile of relative latency.

Setup M6 replicates the workload of five pre-selected APs on two different edge servers, thus providing those APs with enhanced reliability. As a result, server clusters are forced to overlap for those APs, and the geospatial areas of those clusters
are elongated to cover the replicated APs (see Fig. 4). While latencies increase slightly throughout, and the workload balance is marginally worsened, enhancing reliability by replication of a few critical APs does not significantly affect QoS across the deployment.

Illustrating the workloads across the deployment, Fig. A.9 shows the optimal edge server locations resulting from setups (M1–M4), and the individual workloads of servers, some of which may be overlapping in the map illustration. Large workloads appear in densely populated areas, e.g. the city center and the university campus area to the north. The sparsely populated areas have significantly less workload. For scenario M2, the resulting powerful servers are clearly seen. Furthermore, a tighter capacity constraints setup is presented in Appendix. The setup ensures a well balanced workload among servers with standard deviation being the smallest among setups, however the overall latency is worse when compared to M1.

4.3. Large-scale fog deployment

The Fog scenario is based on the Fog computing architecture, which provides a virtualized, hierarchically organized, distributed computing platform, based on small-scale Fog nodes at or near APs, such as common household appliances, smartphones, laptops, and set-top boxes, as well as network infrastructure components on the higher echelons of the hierarchy [15]. The Fog nodes close to UEs are expected to have smaller capacity,
increasing towards the cloud in the infrastructure, forming a “cloud-thing continuum” [59]. Fog thus offers low latency for edge applications in largely isolated deployments, e.g., a business facility. The resulting edge platform infrastructure is based on numerous small-sized clusters with lower capacity servers. Further, where high capacity is needed and increased deployment and management costs are acceptable, a cloudlet could be built with an interconnected set of Fog nodes.

The Fog scenario considers the placement of 150 edge servers, i.e. Fog nodes, with the following capacity constraints: lower limit at 10, and upper limit at 80. The limits are determined with domain expertise: ca. 10 UEs are assumed to be connected to an AP in a typical household, and ca. 80 UEs are assumed to be connected to a more powerful Fog node, covering a building or in an office. Analysis of the original data set indicated that 17 APs had workload, much larger than the upper limit, which necessitates the use of fractional membership.

We compare three different Fog server placement setups:

- **F1** Both lower and upper capacity constraints are present. Fractional membership.
- **F2** No capacity constraints, fractional membership. This corresponds to the discrete k-mean method with fractional membership.
- **F3** Only the upper capacity constraint is present, fractional membership. This was the typical way to balance the workload in the literature.

Quantitative evaluation metrics (see Section 3.3) can be found in Table 5. Fig. 6 further illustrates the resulting optimal edge server locations for these scenarios, while Fig. 7 shows the distributions of workload on the edge servers as histograms for all Fog setups.

**Setup** F1, with both upper and lower capacity constraints, prefers workload balance at the cost of latency. About half of the servers (72) have their workloads at the maximum of 80. The distribution of workload (Fig. 7) is distinctly double peaked, with one peak close to the lower limit and the other close to the upper limit. The reason for the double peaks, is that the workloads tend to be high in the city center, i.e. servers with maximum capacity are needed, whereas in the suburban areas the workloads are close to the minimum (Fig. 6). F1 has a heterogeneous deployment across the area, with low capacity servers fulfilling the latency constraint for the sparse areas, and a dense deployment of high capacity servers in the city center (Fig. 6).

**Setup** F2, conversely, prefers latency at the cost of workload balance. Indeed, with no capacity constraints, an AP is always assigned to the closest server, resulting in the lowest latencies for all the quantities, as well as the mean. However, the setup results in a high variation in server workloads, with 29 servers, or 80% of the total deployment, exceeding the upper constraint of 80 set for other setups. The highest workload is 612, with the distribution heavily skewed to the right (Fig. 7). The servers with the heavy loads are concentrated in the city center and the university campus area to the north, with the low-capacity servers populating the sparse areas (Fig. 6).

Due to the low value of the lower capacity constraint, setup F3 results in a very similar placement candidate as F1. Fig. 5 suggests a tendency towards more shared workload in the dense regions and less in the sparse regions, when compared with F1.

### 5. Discussion

The MEC and Fog scenarios and their various setups illustrate the effect of the latency–workload trade-off, and how to control it, as well the effects of the workload sharing with fractional membership, workload replication, and location preference. While the evaluation used a single data set, it was based on a real-world Wi-Fi setup, and the AP and workload densities in the dataset varied immensely. Arguably, the heterogeneity of the data set guarantees the generalizability of the results.

Indeed, the trade-off between workload and latency can clearly be observed in the results. Using both upper and lower capacity constraints results in a well-balanced load across a heterogeneous data set, but with variance in the resulting latencies, manifesting as highly variable spatial coverage of the AP clusters. Conversely, no capacity constraints result in excellent latency, with AP clusters of equal spatial coverage, but high variance in the resulting workload on the edge servers.

For MEC, fractional membership does not appear to have a strong effect on the resulting deployment. Regarding latency as a measure for QoS, the placement without capacity constraints produces the best results in this respect as it ignores load balancing. If only the upper capacity constraint is set, latency becomes worse for the lower quantities, while workload balance is much improved. Adding also the lower capacity constraint, the workload balance is somewhat improved, while latency slightly worsens. Alternatively, allowing diversity in the capacity of the edge servers may help in reducing the unused capacity typically caused by applying only the upper capacity limit. Further, when a small number of critical servers are assigned a higher location preference or provided high reliability, the effect is minimal on overall latency and workload balance.

The Fog scenario results give three major insights on the hardware configurations and their capacities needed for the Fog nodes, helping to determine the required budget for the Fog deployment. First, omitting capacity constraints in the Fog scenario leads to unrealistically high workloads on some Fog nodes. Capacity constraints are thus necessary at the lowest layer in the Fog hierarchy. Second, for dense areas, with many users and APs, a deep hierarchy of nodes should be deployed, with increasingly capable nodes on the higher levels. On the other hand, in sparser populated areas, it is sufficient to have a flat hierarchy, with a high number of less capable nodes. Third, for the constrained setups in the Fog scenario, depending on the deployment budget, two levels of Fog node hierarchy would be sufficient, as two peaks were seen in the results: one level of light servers for the peak in the lower workloads, and another, more capable one for the peak

---

**Table 5**

Evaluation results of the three Fog setups, comparing relative latency between edge servers and APs, and workload balance on the edge servers. k = number of servers. frac = fractional membership.

| Setup | k | frac | cap | Relative latency | Workload |
|-------|---|------|-----|------------------|----------|
|       |   |      |     | Mean | 25% | 50% | 75% | 95% | S.D. | Min | Max |
| F1 150 | yes | up | low | 0.05 | 0   | 0   | 0.02 | 0.26 | 28   | 10  | 80  |
| F2 150 | yes | no | 0.02 | 0   | 0   | 0.09 | 77   | 1   | 612  |
| F3 150 | yes | up | 0.05 | 0   | 0   | 0.02 | 0.28 | 29   | 1   | 80  |
on the higher workloads. This simplifies the deployed Fog node configurations.

The possibility to share the workloads between edge servers makes the PACK available for wide variety of deployment scenarios. In Fog, i.e. dense deployments of low capacity servers, the benefit of this relaxation is emphasized when the workload of an existing AP is greater than the capacity limit of the associated edge server(s). This way, partial workload can be handled in premises, while the rest is horizontally shared.

In the MEC scenario, shared workload did not have such an effect. This is due to the distribution of the workloads of APs: the average workload in sparse regions is relatively small compared to the predefined capacity range, which reduces the need for sharing. Indeed, fractional membership is more useful when tighter capacity limits are applied, and the average workload in sparse areas is larger. In such cases, finding an optimized solution with hard membership takes more computational time and, in some cases, a solution may not even be found.

Fig. 5. APs with shared workloads in fractional membership scenarios. Black: the AP is assigned to exactly one server. Red: the workload of the AP is shared across servers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Adding to the practical deployment parameters of PACK, location preference is an option in real-world deployments. Certain facilities may require a server, e.g. a research facility, that may or may not be shared with the users of the mobile network. Such prefixed locations should be taken into account in finding the optimal solutions, since the effects are seen in placing of the other servers, and on how the workload is balanced. Such servers could be left out of the initial deployment altogether, and later used as a reserve capacity.

Further, location preference can be considered from the critical network infrastructure point-of-view, where selected APs are given more weight in the placement algorithm. This approach demonstrates an easy but effective solution for addressing location preference in placement algorithms. Furthermore, it enables one to easily allow several preference classes with different weights.

Similarly, reliability is a considerable pragmatic challenge in edge computing in general. As demonstrated, with PACK and in the previous work, a way to address reliability is to add duplicate capacity that becomes operational in case of server failures. Naturally, such an approach further increases initial deployment, maintenance and management costs. In placement algorithms, a feasible option, as demonstrated, is to define a set of servers as a part of the critical infrastructure, and provide duplicate capacity only for them.
Overall, edge server placement is a problem of increasing importance in future distributed IoT systems. We believe that the PACK algorithm, with extensive set of deployment parameters, based on significantly large data set and detailed statistical evaluation in a multiple scenarios, simultaneously and comparably, is a noteworthy addition to the knowledge in the field.

6. Conclusion

Solving the edge server placement problem provides the foundation for further study of other edge resource provisioning problems, such as online load balancing. In this paper, we first conducted a survey of existing physical edge server placement algorithms and identified an extensive set of parameters utilized in placement. Based on the findings, we presented the PACK algorithm that considers the parameter set in optimal placement for edge servers with load balancing under capacity constraints, while minimizing latencies between the servers and their associated APs.

We evaluated the PACK algorithm, and studied the effects of different placement parameters, with simulations in two distinct edge deployment scenarios, corresponding to MEC and Fog computing. The simulations were based on a large real-world data set, collected in a city-wide Wi-Fi network between the years 2007–2015.

The evaluations provided numerical statistics on QoS and QoE for analyzing the performance of PACK in each scenario. The latency–workload balance was illustrated with extensive examples, and methods for controlling the trade-off were detailed. The influence of workload sharing, workload replication and location preference on the placement were also discussed in detail.

Further, the evaluations provide broad insights on how well the algorithm performs with different architectural design options and on planning of deployment setups with a set of constraints. Edge computing systems are developing towards large numbers of distributed, small capacity computational units, deployed into individual sites, as in Fog computing. Moreover, in evaluation, a number of pragmatic challenges for mobile network operators were taken into account, providing insights on how well the algorithm performs under real-world constraints.

We argue that the flexibility of the PACK algorithm, combined with the easy to use open-source R based software package, provides an analytical toolbox to study different edge server placement schemes, before making the actual deployment decisions. Such a tool box is a key component towards the future vision of edge-native artificial intelligent solutions [36], providing the theoretical background to find optimal solutions to the placement problem, with real-world edge applications and fine-grained data collected on their performance.

CRediT authorship contribution statement

**Tero Lähderanta:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Teemu Leppänen:** Conceptualization, Methodology, Validation, Writing - original draft, Writing - review & editing, Funding acquisition. **Leena Ruha:** Conceptualization, Methodology, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Lauri Lovén:** Conceptualization, Methodology, Software, Validation, Data curation, Resources, Writing - original draft, Writing - review & editing, Visualization, Project administration, Funding acquisition. **Erkki Harjula:** Conceptualization, Writing - review & editing, Funding acquisition. **Mika Ylianttila:** Conceptualization, Supervision, Funding acquisition. **Jukka Riekki:** Conceptualization, Supervision, Funding acquisition. **Mikko J. Sillanpää:** Conceptualization, Methodology, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by Academy of Finland 6Genesis Flagship (grant 318927), the Infotech Oulu research institute, Finland, the Future Makers program of the Jane and Aatos Erkko Foundation, Finland and the Technology Industries of Finland Centennial Foundation, by Academy of Finland Profi 5 funding for mathematics and AI: data insight for high-dimensional dynamics (grant 326291), and the personal grant for Lauri Lovén on Edge-native AI research by the Tauno Tonning foundation, Finland.

Appendix. Scenario with tight capacity limits

In this section we consider a MEC placement scenario where both lower and upper capacity constraints are set. We set a relatively tight width for the capacity, resulting in a lower capacity...
Table A.6
Comparing the distance and balance of the nine different deployment scenarios (see Section 4.2). k = number of servers. frac = fractional membership. cap = capacity constraints (up = upper constraint, low = lower constraint). Mean = the average AP distance weighted by the workload. \( \alpha \% \) = the distance within which \( \alpha \% \) of the workload is from the associated server. See Section 3.3 for further information. The workload balance is measured with the standard deviation of the server workloads (S.D.) and the minimum (Min) and the maximum (Max) obtained server workload. The green background indicates the best performance in terms of the different quality measures in the MEC scenarios M1 - M6 and the tight capacity limits scenario.

| Setup   | k | frac | cap       | Relative latency | Workload |
|---------|---|------|-----------|------------------|----------|
|         |   |      |           | Mean 25% 50% 75% 95% | S.D. Min Max |
| M1      | 20| no   | up/low    | 0.59 0.06 0.28 0.81 2.11 | 90 327 526 |
| M2      | 20| no   | no        | 0.40 0.02 0.29 0.48 1.33 | 336 55 1013 |
| M3      | 20| no   | up        | 0.55 0.10 0.38 0.71 1.61 | 162 57 526 |
| M4      | 20| yes  | up/low    | 0.59 0.04 0.28 0.70 2.11 | 89 327 526 |
| M5      | 20| no   | up/low    | 0.59 0.10 0.28 0.73 2.11 | 89 327 526 |
| M6      | 20| no   | up/low    | 0.63 0.10 0.30 0.85 2.19 | 95.2 327 526 |
| Tight   | 20| no   | up/low    | 0.63 0.04 0.30 0.75 2.25 | 41.9 377 476 |

Fig. A.8. Placement of 20 MEC servers with tight capacity limits. Crosses denote the server locations. WI: workloads of the servers. The color shades visualize the server regions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In terms of latency, the tight capacity limits scenario performs similarly to the scenario M6 as can be seen in the mean value in Table A.6. Furthermore, the scenario has the worst-case latency quantile (95%). This is confirmed in Fig. A.8, where the server regions are large in the sparse regions. However, it is clear that the tight capacity limits ensure a well balanced workload with approximately 0.5 times smaller standard deviation than in the other capacitated scenarios in Table A.6.
Fig. A.9. Assigned workload of each MEC server. Black dots indicate the APs and colored dots represent the servers, the shade of color indicating the workload ($W_l$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] M. Ackerman, S. Ben-David, S. Brânzei, D. Loker, Weighted clustering, in: Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012.

[2] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, in: SODA'07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, pp. 1027–1035, URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

[3] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, B. Lepri, A multi-source dataset of urban life in the city of milan and the province of trentino, Sci. Data 2 (150055) (2015).

[4] M. Berkelaar, et al., lpSolve: Interface to lp.solve v. 5.5 to solve linear/integer programs, 2015, R package version 5.6.13, URL https://CRAN.R-project.org/package=lpSolve.

[5] D. Bhatta, L. Mashayekhy, Generalized cost-aware cloudlet placement for vehicular edge computing systems, in: 2019 IEEE International Conference on Cloud Computing Technology and Science, CloudCom, 2019, pp. 159–166.

[6] K. Bilal, O. Khalid, A. Erbad, S.U. Khan, Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers, Comput. Netw. 130 (2018) 94–120.

[7] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.
[8] S. Borgwardt, A. Brieden, P. Gritzmann, An LP-based k-means algorithm for balancing weighted point sets, European J. Oper. Res. 263 (2) (2017) 349–355.
[9] M. Bouet, V. Conan, Mobile edge computing resources optimization: a geo-clustering approach, IEEE Trans. Netw. Serv. Manag. 15 (2) (2018) 787–796.
[10] J. Brimberg, P. Hansen, N. Mladenovic, S. Salhi, A survey of solution methods for the continuous location-allocation problem, Int. J. Oper. Res. 5 (2008) 1–12.
[11] L. Chen, J. Wu, G. Zhou, L. Ma, QUICK: QoS-guaranteed efficient cloudlet placement in wireless metropolitan area networks, J. Supercomput. 74 (8) (2018).

[30] T. Lähderanta, L. Lovén, L. Ruha, rpack: R package for capacitated clustering, J. Parallel Distrib. Comput. (2020), https://dx.doi.org/10.1016/j.jpdc.2020.05.002.
[31] L. Kaufman, P. Rousseeuw, Clustering by Means of Medoids, in: Delft University of Technology: reportsoftheFacultyofTechnicalMathematics and Informatics, Faculty of Mathematics and Informatics, 1987.

[25] S. Kang, R. Linna, S. Guo, W. Li, X. Qiu, Geographic clustering based coveragelocationmodelswithQ-coveragerequirement, Procedia Eng. 149 (2016) 475–494.
[26] M. Karatas, N. Razi, H. Tozan, A comparison of p-median and maximal coveragerequirements, Annals of Operations Research 278 (1) (2019) 1–24.

[40] N. Mohan, A. Zavodovski, P. Zhou, J. Kangasharju, Anveshak: Placing edge servers in the wild, in: Proceedings of the 2018 Workshop on Mobile Edge Computing, ACM, United States, 2018, pp. 7–12.
[41] R. Z. Farahani, M. Hekmatfar, Facility location: concepts, models, algorithms and case studies, in: Contributions to Management Science, Physica-Verlag, Heidelberg, 2009.

[52] S. Wang, Y. Zhao, L. Huang, J. Xu, C.-H. Hsu, QoS prediction for service allocation at the cloud data center, IEEE Trans. Serv. Comput. 12 (2019) 160–168.

[60] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, J. Optim. Theory Appl. 105 (2000) 475–494.
[61] E. Wang, D. Li, B. Dong, H. Zhou, M. Zhu, Flat and hierarchical system deployment for edge computing systems, Future Gener. Comput. Syst. 105 (2020) 308–317.
[62] S. Wang, Y. Zhao, L. Huang, J. Xu, C.-H. Hsu, QoS prediction for service recommendations in mobile edge computing, J. Parallel Distrib. Comput. 127 (2019) 134–144.
[63] S. Wang, Y. Zhao, J. Xu, J. Yuan, C.-H. Hsu, Edge server placement in mobile edge computing, J. Parallel Distrib. Comput. 127 (2019) 160–168.

[72] X. Guan, X. Wan, T. Wang, Y. Li, A long-term cost-oriented cloudlet planning method in wireless metropolitan area networks, Electronics 8 (11) (2019) 1213, URL https://www.mdpi.com/2079-9292/8/11/1213.
[73] H. Sinky, B. Khalfi, B. Hamdaoui, A. Rayes, Adaptive edge-centric cloud computing, Softw. - Pract. Exp. 50 (2020) 177–183.
[57] H. Yao, C. Bai, M. Xiong, D. Zeng, Z. Fu, Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing, Concurr. Comput.: Pract. Exper. 29 (16) (2016).
[58] H. Yin, X. Zhang, H.H. Liu, Y. Luo, C. Tian, S. Zhao, F. Li, Edge provisioning with flexible server placement, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1031–1045.
[59] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niaakanlaihiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. (2019).
[60] F. Zeng, Y. Ren, X. Deng, W. Li, Cost-effective edge server placement in wireless metropolitan area networks, Sensors 19 (1) (2018).
[61] G. Zhou, J. Wu, L. Chen, G. Jiang, S.-K. Lam, Efficient three-stage auction schemes for cloudlets deployment in wireless access network, Wirel. Netw. 25 (6) (2019) 3335–3349.

Tero Lähderanta is a Ph.D. student at the Research Unit of Mathematical Sciences, University of Oulu. He received his M.Sc. degree in statistics in 2018 from the University of Jyväskylä. Currently, he is participating in a project that is related to distributed data analysis in the IoT. His research interests include clustering methods, Markov chain Monte Carlo algorithms and Bayesian methods in general.

Teemu Leppänen is a university lecturer with the Center for Ubiquitous Computing, University of Oulu, Finland. His research interests include Internet of Things edge and mist computing with focus on distributed artificial intelligence. Currently he participates in the 6Genesis Flagship, an eight-year research program to develop fundamental concepts of 6G networks. He has visited the University of Tokyo, Japan, in 2012-13 and the University of Calabria, Italy, in 2019. He has authored over 50 peer-reviewed articles in international journals, research books and conferences, where he has received three best paper awards. He is Senior Member of IEEE.

Leena Ruha (née Pasanen) works as a senior scientist at the Natural Resources Institute Finland. She received her Lic.Phil. degree in statistics in 2006, and her Ph.D. degree in applied mathematics in 2012, both from the University of Oulu. Her research interests include development and application of statistical and mathematical methods for e.g. distributed systems and population dynamics. She has authored 20 peer-reviewed articles.

Lauri Lovén, M. Sc. (Stat), is a senior member of IEEE and a doctoral candidate at the Center for Ubiquitous Computing (UBICOMP), University of Oulu in Finland. With a background in mathematical statistics and ICT business, his current research concentrates on distributed AI, machine learning and data analysis methods and applications in edge computing. He has co-authored 2 patents and ca. 30 research articles in international peer-reviewed journals, conferences and workshops.

Mikko J. Sillanpää is a full professor in statistics in the Research Unit of Mathematical Sciences, University of Oulu. He has a long and acknowledged experience in Bayesian statistics and computational methods in high-dimensional problems in biology, medicine and other fields. His research interests cover e.g., variable selection, functional data analysis and predictive analytics. He is a Director of the University of Oulu HiDyn program 2019-2023 funded by the Academy of Finland PROFS: multidisciplinary data science and artificial intelligence hub strengthening all of University of Oulu’s profiling areas and the 6G Flagship.

Erkki Harjula is an Assistant Professor of Wireless System Level Architecture for Future Digital Health-care at University of Oulu, Finland. He got a D.Sc. (Tech) degree in 2016 and M.Sc. (Eng) degree in 2007 from University of Oulu. Dr Harjula’s research interests include IoT communication, edge computing, distributed computing, distributed AI and distributed ledger technologies. In research, he is mostly concentrated in intelligent trustworthy distributed IoT and edge computing/computation in future digital healthcare. He has published more than 70 research articles, cited by more than 1200 times, and has an h-index of 18 (Google Scholar). He is an editorial board member in Springer Wireless Networks (WINE) - Journal of Mobile Communication, Computation and Information, and has acted as a Reviewer for several international scientific journals. He visited Columbia University in NY, USA, in winter 2008-2009. He has supervised numerous undergraduate and doctoral students and led several research projects. He is also a member of IEEE.

Mika Ylianttila (M. Sc, Dr.Sc, eMBA) is a full-time associate professor (tenure track) at the Centre for Wireless Communications - Networks and Systems research unit, at the Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, Finland. He is the director of Communications Engineering Doctoral Degree Program and he leads NSOFT (Network security and softwarization) research group which studies and develops secure, scalable and resource-efficient techniques for 5G and beyond 5G and IoT systems. He has co-authored more than 190 international peer-reviewed articles. He is a Senior Member of IEEE and associate editor in IEEE Transactions on Information Forensics and Security.

Jukka Riekki is a full Professor of Software Architectures for Embedded Systems at the University of Oulu. He is the Dean of the Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland, and one of the PI’s in the Center for Ubiquitous Computing. He holds a Dr.Sc.Tech. in Technology from the University of Oulu and is the author of over 200 peer-reviewed articles. His main research interests are in edge computing, edge AI and intelligent systems. He is actively involved in the 6G Flagship program, in developing Fab Lab Oulu, and developing and applying a new agile research-to-business model in the Allied ICT Finland program.

Journal of Parallel and Distributed Computing 153 (2021) 130–149