Scaling up an Edge Server Deployment

Lauri Lovén*, Tero Lähderanta†, Leena Ruha†, Teemu Leppänen*, Ella Peltonen*, Jukka Riekki*, and Mikko J. Sillanpää†

* *Center for Ubiquitous Computing*
University of Oulu
Oulu, Finland
{first.last}@oulu.fi

†*Research Unit of Mathematical Sciences*
University of Oulu
Oulu, Finland
{first.last}@oulu.fi

***Abstract*—In this article, we study the scaling up of edge computing deployments. In edge computing, deployments are scaled up by adding more computational capacity atop the initial deployment, as deployment budgets allow. However, without careful consideration, adding new servers may not improve proximity to the mobile users, crucial for the Quality of Experience of users and the Quality of Service of the network operators. In this paper, we propose a novel method for scaling up an edge computing deployment by selecting the optimal number of new edge servers and their placement, and re-allocating access points optimally to the old and new edge servers. The algorithm is evaluated with two scenarios, using data on a real-world large-scale wireless network deployment. The evaluation shows that the proposed method is stable on a real city-scale deployment, resulting in optimized Quality of Service for the network operator.**

***Index Terms*—edge computing, facility location, service scaling**

I. Introduction

The modern smart devices, from smartphones to home IoT, from industrial applications to smart cities and smart transportation, are ushering in an era of pervasive computing. The massive amounts of data generated by these devices provide a ground for various novel applications, while also introducing novel challenges for data processing and connectivity [1]. Indeed, current cloud computing systems and network infrastructures may not provide enough computing capacity to manage latency and performance requirements set by the modern pervasive computing systems [2].

Edge computing refers to technologies and development methodologies for distributing and running computations close to the user devices, ”on the edges of the network”, at the network infrastructure devices or dedicated local systems providing computing resources for the user devices. Such computations typically include the collection and preprocessing of application-specific multimodal data [3], [4] or facilitating real-time user interactions such as augmented or virtual reality applications [5]. The expected advantages of edge computing include low latency and high bandwidth between user devices and edge components, crucial for real-time applications [6], support for user mobility [7], [8], and increased privacy especially with applications relying on highly personal data [9].

Edge computing platforms aim to provide location-aware services and multitenancy of applications at the close physical and logical proximity. A number of solutions have already been proposed, e.g. cloudlets [10] and Fog computing [11], or even Kubernetes-based over-the-air LTE/5G Edge DevOps platforms [12], that illustrate different technologies and implementation strategies [13].

In addition to placing data and computations [1], placing computational resources [14] is a key issue in edge computing. Edge computing requires a flexible and scalable deployment of edge servers to support user mobility, the inherent dynamicity of the operational environment, and the variety of applications, some requiring real-time responsiveness. The deployments are often geographically large, spanning entire cities. The edge servers are often deployed densely, based on observations on application workload and usage patterns and the resulting network traffic. Often, the available deployment budgets lead to homogeneous hardware in the edge servers, which may result in under- or overused capacity. The simplest approach is to provide a clustered deployment without initially considering capacity or proximity constraints. However, the operators are expected to ensure efficient resource usage as well as a sufficient Quality of Experience (QoE) for the users, while maintaining overall QoS (Quality of Service) in their system. Therefore, when planning the deployment architecture, promixity to the mobile users is a crucial element.

Edge-based service and functionality availability is constantly increasing, and the soon-to-come 5G networks will only further increase their availability. Further, the increased usage of smartphones, other personal smart devices, connected vehicles, autonomous devices, and other novel technologies lead to an unforeseen variety of edge applications and services with increasing requirements for computation and communication. As a result, edge server deployments need to be scaled up to meet the increasing demands. Scaling up an edge server deployment, the operator needs to find the optimal number of new edge servers, their optimal locations, and the allocations of Access Points (APs) to these servers.

In this paper, we present the following novel contributions:

1) We present a novel method for scaling a deployment, finding the optimal number of new edge servers, placing

them in an existing edge network, and allocating APs for the edge servers.
2) We evaluate our method with three scenarios which correspond to real-life use cases for a network operator. The evaluation indicates that the proposed method results in optimized QoS for the network operator.
3) We present the results based on a real-world data set of a smart city Wi-Fi usage, collected over nine years and comprising hundreds of millions of connections.

II. Related Work

In the previously proposed placement schemes, the best proximity, with regard to some metric, was often the initial assumption in the deployment planning [14]. When the aim is to address minimized proximity, upgrading server capacity to the existing (co-located) deployment is not sufficient as distances may not be optimized for the added users and their workload. Moreover, if server capacity is fixed, the solution prohibits over-provisioning and QoE cannot be guaranteed. Also, on-demand provisioning has been considered, e.g. [15], [16], that increases scalability in response to the online workload. But such efforts may be difficult to realize by the network operators.

The number of new servers may be dictated by a set budget. Sometimes it is necessary to determine what is the number of servers that would produce a best trade-off between the budget and QoS. To explore this trade-off, some previous work [17], [18] evaluate the average user latency as a function of the number of edge servers.

The survey of Lähderanta et al. [14] studies edge server placement extensively. For example, in the study by Wand et al. [17] a fixed number of edge servers are placed by minimizing the geospatial distances while concurrently seeking for a balanced workload distribution. In [18] a hierarchical tree-like structures are used to locate fixed number of edge servers without capacity limits. Yin et al. [19] propose a heuristic decision-support management system for server placement that enables the discovery of unforeseen server locations. Guo et al. [20] place a fixed number of edge servers in a two-step scheme, where first the servers are located using the k-means algorithm and then the APs are allocated to the servers with the aim to minimize the communication delay and to balance the workload.

The method presented in this paper is based on the capacitated location allocation method, presented in a previous study of ours [14]. To the best of our knowledge, however, no articles have studied the scaling up of an existing edge server deployment.

III. Methods

A. Placement model

We base our method for the placement of new edge servers on the PACK algorithm, proposed in our earlier work [14], detailed below. The algorithm finds optimal locations for a fixed number of edge servers, minimizing the geospatial distances to APs and satisfying the capacity constraints. Such an optimization problem can be considered as a capacitated location allocation problem [21]–[24] and more specifically as a capacitated p-median type problem [24].

In a p-median problem typically the Euclidean distance is used. However, PACK applies the squared Euclidean distances producing k-means type clustering with spherical-like clusters with centralized cluster heads [25]. This results in a star-like topology with spatially centralized edge servers for both dense and sparse areas, contributing towards better proximity, i.e. QoS, particularly in the remote APs. Thus, the approach can actually be considered 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-medoid method [26]. Given a data set of n APs, let x_i be the coordinates a for AP i and w_i the corresponding workload. In practice, workload w_i is determined by the maximum amount of simultaneous users in AP i.

Let us denote by c_j, $j = 1, 2, ..., k$, the coordinates for k edge servers and by y_{ij} the membership of AP i to the edge server j.

Our aim is to optimize the locations of the servers and the allocations of the APs by minimizing the squared Euclidean distance between the edge servers and the APs they cater, weighted by the workload of each AP, while taking into consideration the capacity constraints of each server.

More specifically, the objective function to be minimized is

$$\underset{c_j, y_{ij}}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{k} w_i ||x_i - c_j||^2 y_{ij}. \quad (1)$$

The optimization is carried out with the following constraints:

$$c_j \in \{x_1, x_2, \dots, x_n\} \quad \forall j, \quad (2)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j, \quad (3)$$

$$\sum_{j=1}^{k} y_{ij} = 1 \quad \forall i, \quad (4)$$

$$L \leq \sum_{i=1}^{n} w_i y_{ij} \leq U \quad \forall j. \quad (5)$$

These constraints correspond to the following assumptions: An edge server must be co-located with an AP (2), each AP is connected to exactly one edge server (3), (4), and the total workload of the APs connected to one edge server can be uniformly distributed between a lower (L) and an upper (U) limit (5).

The optimization problem is NP-hard, calling for approximate solutions. PACK [14] is an iterative block-coordinate descent algorithm, detailed in Alg. 1, consisting of two main steps: the allocation-step on line 4 and the location-step on line 5. PACK iterates these two steps until the locations of edge servers c_j do not change. However, this type of iteration does not guarantee that the result is the global minimum. Therefore PACK runs with N initial values for the server locations, which

Algorithm 1 PACK-algorithm [14]

Input: x_i, w_i, k, N

Output: Edge server locations c_j^* and allocations $y_{ij}^*, j = 1, \ldots, k$

```
1: for $i = 1$ to $N$ do
2:     Initialize $c_j$, $j = 1, 2, \ldots, k$ 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:         $S \leftarrow$ the value of the objective function
7:     end while
8:     if $S < S_{min}$ or $i = 1$ then
9:         $S_{min} \leftarrow S$
10:        $c_j^* \leftarrow c_j$
11:        $y_{ij}^* \leftarrow y_{ij}$
12:    end if
13: end for
14: return $c_j^*, y_{ij}^*$
```

Algorithm 2 sPACK-algorithm

Input: $x_i, w_i, k_1, k_2, N, c_{j_{\text{old}}}, j_{\text{old}} = 1, \ldots, k_1$

Output: New allocations for all edge servers $y_{ij_{\text{all}}}^*$, with $j_{\text{all}} = 1, \ldots, k_1 + k_2$, and locations for the new servers $c_{j_{\text{new}}}^*$, where $j_{\text{new}} = k_1, \ldots, k_2$

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

are obtained using the k-means++ algorithm [27], and selects the placement obtained with iteration with the best objective function value.

B. Scaling PACK

We modify the PACK algorithm to support adding new servers for an existing server network. Consider a deployment where k_1 servers $c_1, \ldots, c_{k_1}$ are previously placed and the aim is to optimally place k_2 more servers $c_{k_1+1}, \ldots, c_{k_1+k_2}$. The placement is optimized by minimizing (1) with respect to $c_{k_1+1}, \ldots, c_{k_1+k_2}$ and $y_{ij_{\text{all}}}$ $\forall\ i = 1, \ldots, n,\ j_{\text{all}} = 1, \ldots, k_1 + k_2$, assuming $c_1, \ldots, c_{k_1}$ fixed. In practice, this means that in Alg. 1, while the allocation step must still consider all APs, the location step should update only the locations of the new servers. The new, scaling PACK algorithm (sPACK) is detailed in Alg. 2.

C. Selection of the number of servers

We select the number of additional servers based on the elbow method, often used in clustering analysis for selecting the optimal number of clusters [28]. In the elbow method, a curve, referred here as cost-effectiveness curve, is drawn with the number of clusters in the horizontal axis and the minimum of the objective function in the vertical axis. The number of clusters is then selected visually as the "elbow" where increasing the number of clusters does not appear to produce considerable decrease in the value of the objective function.

D. Measuring the Quality of Service

Following multiple other studies [14], [16]–[18], [20], we measure QoS by proximity, i.e. the Euclidean distances between the APs and their allocated edge servers. The average AP distance, weighted by the workload, is

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

where $W = \sum_{i=1}^{n} w_i$ corresponds to the total workload of all the studied APs.

Further, following our previous work [14], we take a closer look at the proximity distributions by the sample quantiles q_α that measure the distance within which α proportion of the workload is from the associated edge server. Accordingly, selecting α close to 1 evaluates the worst case QoS, whereas $\alpha = 0.5$ corresponds to the median QoS. The quantiles q_α can be obtained by solving $F(q_\alpha) = \alpha$, where F is the empirical cumulative distribution function of the workload.

IV. Evaluation

A. Data

We test our methods with a real-world Wi-Fi network data set, collected from the PanOULU public network access points (AP) in the city of Oulu, Finland, in 2007–2015 [29]. The raw data contains the timestamps, durations and source identifications of all the connections to the APs during the observation period. Following our earlier work [14], we use only 2014 data, the last full year in the data set. The number of individual Wi-Fi connections that year was 257,552,871 on 450 active access points (AP). The PanOULU APs are shown in Fig. 1.

Each connected user is assumed to introduce a workload of one to an AP and the edge server that AP is connected to. The number of concurrently connected users for one busy AP of a local polytechnic on the first week of September is depicted in Fig. 2. To make sure the edge servers have capacity for even the peak hours, we choose the highest number of

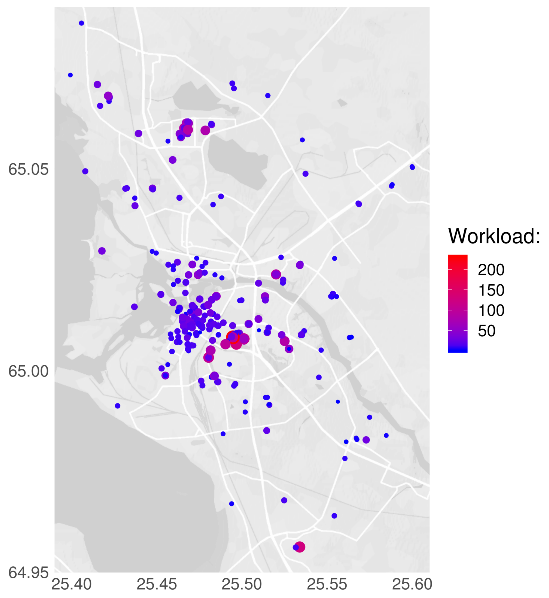


Fig. 1: Locations of the PanOULU access points.

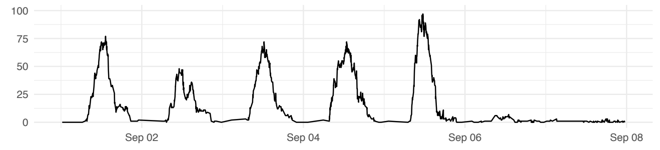


Fig. 2: Connected users on the first week of Sept. 2014 at one of the panOULU access points.

concurrent users in 2014 for each AP as the relative workload the APs introduce on the edge server. Fig. 3 shows that the AP workloads are distributed roughly exponentially, with a small number of high workloads and a fat tail of low workloads.

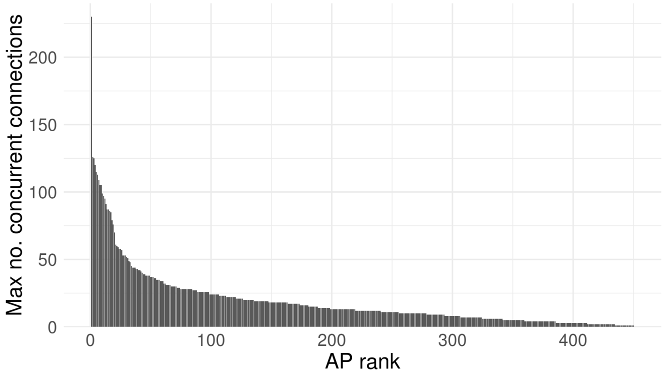


Fig. 3: Workload of the PanOULU access points.

B. Scenarios

We evaluate the proposed method in three scenarios. In two of the scenarios, there is an existing deployment of edge servers, which is subsequently grown using the proposed method to find the number of new edge servers to deploy, as well as their placements. These are compared to a reference scenario which places all servers and allocates their APs optimally, without an intermediate scaling step. The scenarios are described in detail below and summarized in Table I.

TABLE I: Evaluation scenarios.

Scenario	Existing	New	Capacity
Reference	0	15—25	[300,600]
Small	5	10—20	[300,600]
Large	15	0—10	[300,600]

1) Reference deployment: The reference deployment starts from a clean slate. 15–25 edge servers are placed and their AP allocations set using the method proposed in our earlier work [14]. The capacity ranges for edge servers are chosen to be [300,600] to accommodate the workloads of the AP allocations in the range of edge servers: the more edge servers, the fewer APs need to be allocated to each edge server, and the smaller their combined workloads. The center point in the capacity range, 450, divides the total workload of all APs evenly between 20 edge servers.

2) Small deployment: The first scaling scenario assumes there are five deployed edge servers. This corresponds to a small-scale testing deployment in strategic locations, such as the operator R&D center or a university, which is then extended towards a commercial service. In this scenario, further, the operator is assumed to have a flexible budget to add 10 to 20 new edge servers. The capacities are set identical to the reference scenario.

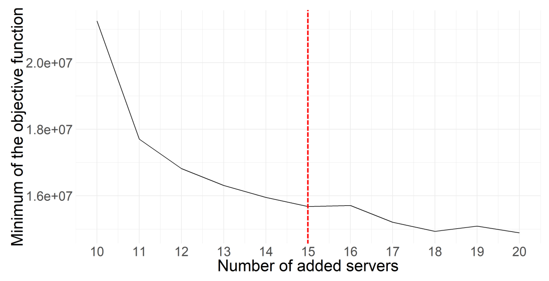
3) Large deployment: The second scenario assumes a setup of 15 existing edge servers, placed in optimal locations using the method proposed by Lähderanta et al. [14], and there are potentially 0–10 new ones to deploy. This corresponds to a business-as-usual scenario where the operator periodically checks if edge application user QoS could be improved by new edge server deployments. The existing deployment as well as the range of new servers to add in the Large scenario are chosen to reflect the other two scenarios to make them comparable.

C. Results

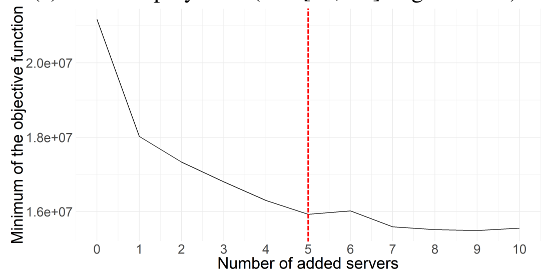
For the Small Deployment, with 5 edge servers placed by a domain expert, the cost-effectiveness curve in Fig. 4 shows an elbow at the midpoint of 15 added servers. The resulting APs and their allocated edge servers are depicted in panel (a) of Fig. 5, with the fixed edge servers shown as asterisks, the new servers as X's, and the APs allocated to particular server all colored identically within a convex hull.

For the Large Deployment, with 15 existing edge servers placed in optimal locations, the cost-effectiveness curve suggests deploying 5 extra servers. These were placed as indicated in the right panel of Fig. 5.

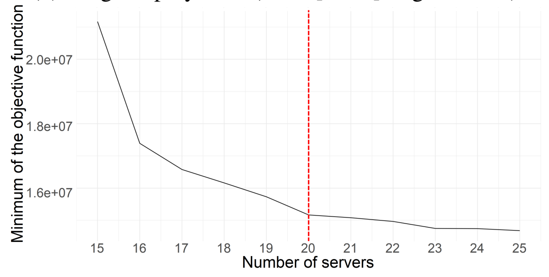
Table II demonstrates how the QoS measures differ between the scenarios and a reference deployment of 20 optimally placed edge servers. The reference scenario, as expected, gives the best objective function minimum and QoS on all the



(a) Small deployment (5 + [10, 20] edge servers)



(b) Large deployment (15 + [0, 10] edge servers)



(c) Reference deployment ([15, 25] edge servers)

Fig. 4: Cost-effectiveness curves of Small, Large and Reference deployment scenarios.

measures. The Small deployment scenario performs better than the Large deployment scenario in terms of the objective function minimum. However, in terms of QoS, the Small deployment scenario excels in mean proximity as well as in the 75% quantile, while the Large scenario has better 25% quantile, median and worst case behavior, those being equal or nearly-equal to the reference.

TABLE II: EVALUTION RESULTS FOR THE THREE SCENARIOS.

		Proximity (km)				
Scenario	**Obj. function**	**Mean**	**25%**	**50%**	**75%**	**95%**
Reference	15.2e+06	0.556	0.05	0.29	0.65	2.10
Small	15.7e+06	0.571	0.11	0.31	0.62	2.14
Large	15.9e+06	0.593	0.05	0.29	0.75	2.11

V. DISCUSSION

We compared the scaled Small and Large deployment scenarios with the reference scenario which had no scaling. The Small deployment scenario turned out to be close to the reference scenario in terms of the objective function and the mean proximity, whereas the Large deployment scenario produced inferior results. However, the results are highly dependent on the placement of the initial edge servers before scaling. Indeed, the initial placement affects the placement of new servers. In this case, both of the scaling scenarios had a small number of servers placed non-optimally, which overall causes worse QoS than in the reference scenario.

For all the scenarios, the total number of servers turned out to be 20, with the number of new servers at the midpoint of the budget range. This is unsurprising considering the capacity limits in the scenarios: the midpoint of the capacity range was 450, and the total workload of all AP's divided by 450 is 20. Indeed, in clustering, the objective function typically decreases while the number of clusters (here edge servers) is increased. However, as we apply both the lower and the upper limits to the capacity, increasing the number of servers may not always decrease the objective function: if a high number of servers is placed, the algorithm has to make spatial compromises for fulfilling the lower capacity limit, leading to higher (i.e. worse) objective function values. Similarly, if a low number of servers is placed, obeying the upper capacity limit is difficult. Due to these effects, the elbow approach actually appears to have a tendency to favor the number of servers that divides the whole workload to the midpoint of the capacity limits.

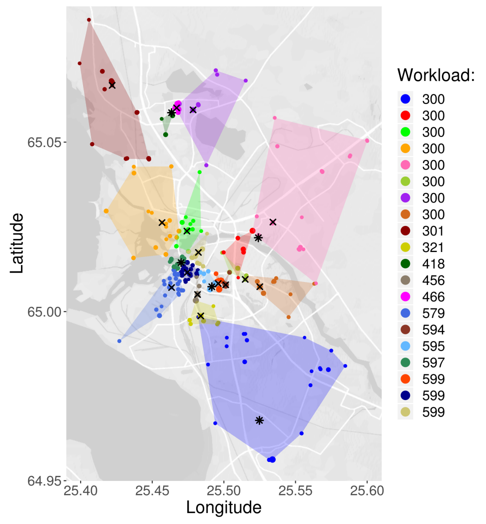
Scaling up the deployments occasionally produces allocations which overlap spatially. For example, in Fig. 5, both the Small and Large deployments have an edge server from the original deployment (marked with an asterisk) within the north-easternmost cluster, with some of the nearby APs allocated to another edge server (marked with an X). Such deployments are not obvious and are less likely to occur when a domain expert is placing the servers, and speak in favor of the proposed method.

On the other hand, spatial overlap seems to occur due to mistakes made in the original placement. Indeed, in Fig. 7 (a) and (b), the red boxes illustrate how the edge server placed in the original deployment is different from the optimal placement in the reference scenario. In such cases, comparing with the reference, the network operator could consider the replacement of some of the original edge servers to further improve QoS.

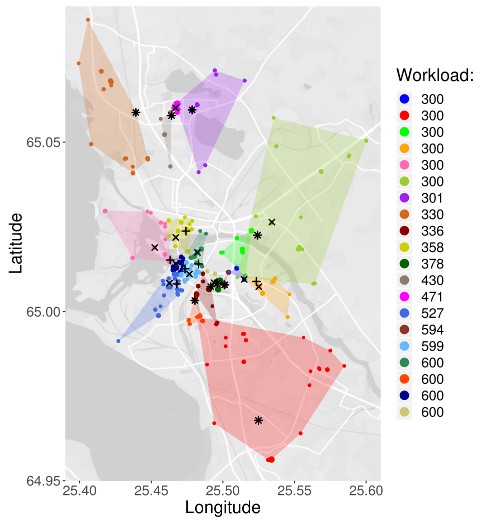
Typically in the edge server placement literature, the servers are co-located with APs. However, omitting this constraint (2) may provide valuable information about new candidate places where APs may be located. The domain expert could then detect, close to these optimal locations, new candidate places where a new server could be set. Then the algorithm could be re-run with the co-location constraint (2) with the newly discovered places added to the candidate places, and the QoS improvement considered. This procedure resembles the approach taken by Yin et al. [19].

Overall, both scenarios appeared to produce relatively good QoS compared to the reference scenario. The proposed algorithm works well both in a case where a small pilot deployment is extended to an extensive server network, and in a case where an existing server network is further expanded.

Further, the proposed algorithm provides an important tool for domain experts to design and analyze different solutions for edge server deployments. This is illustrated by Fig. 7,



(a) Small deployment (5 + 15 servers).



(b) Large deployment (15 + 5 servers).

Fig. 5: Edge server placements and AP allocations with the Small and Large deployment scenarios.

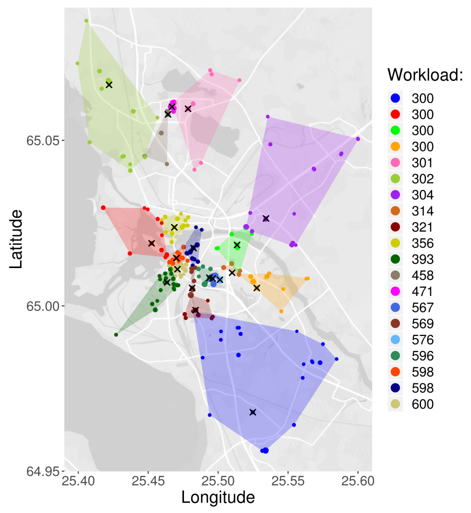


Fig. 6: Edge server placement for Reference deployment (20 servers).

showing the scaled-up Small and Large deployments compared with the edge server placements of the reference deployments. Such information is important for making placement decisions in-situ.

VI. Conclusion

Edge server deployments need to scale up along the growth of edge application usage. In this paper, we presented a novel method for finding the optimal number of new servers to add, their optimal locations, as well as the optimal allocations of the access points for both the old and the new edge servers. We evaluated the method with a real-world data set of Wi-Fi access logs, comparing two example scenarios to a reference setup.

The evaluation showed that the proposed method is stable in city-wide placement scenarios with small and large initial edge server deployments, resulting in optimized QoS for the network operator. As future work we will study, in particular, the capacity constraints and their effect on the optimization process.

VII. Acknowledgements

This research is supported by Academy of Finland 6Genesis Flagship (grant 318927), the Infotech Oulu research institute, the Future Makers program of the Jane and Aatos Erkko Foundation 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, and by the personal grant for Lauri Lovén on Edge-native AI research by the Tauno Tönning foundation.

References

[1] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker, “Context-aware data and task placement in edge computing environments,” in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom.* IEEE, 2019, pp. 1–10.

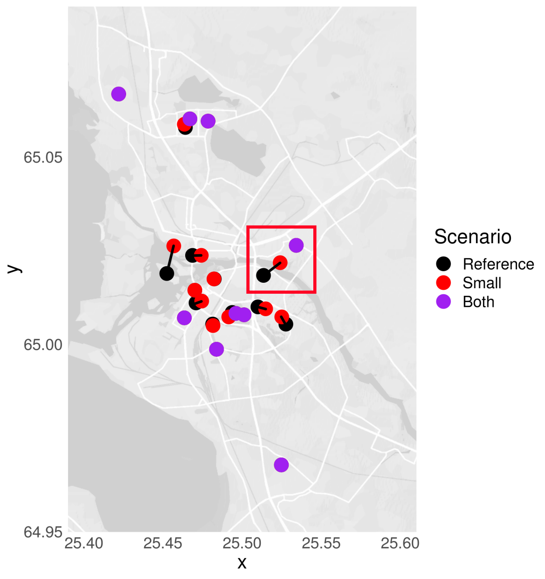
[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, 2019.

[3] L. Lovén *et al.*, “Mobile road weather sensor calibration by sensor fusion and linear mixed models,” *PLoS ONE*, vol. 14, no. 2, pp. 1–17, 2019.

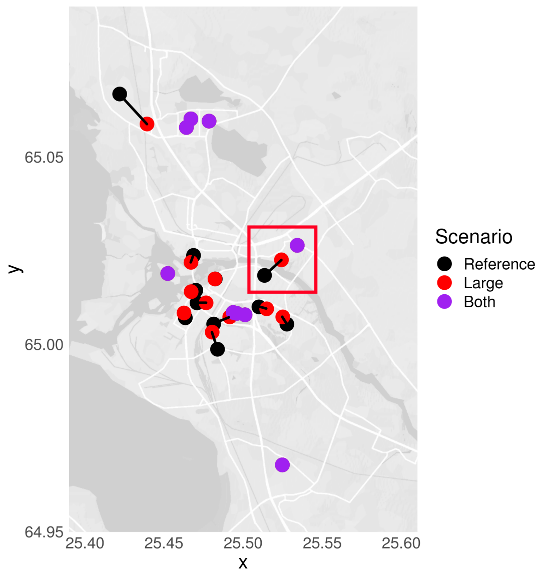
[4] ——, “Towards EDISON: An edge-native approach to distributed interpolation of environmental data,” in *28th International Conference on Computer Communications and Networks (ICCCN2019), 1st Edge of Things Workshop 2019 (EoT2019).* Valencia, Spain: IEEE, 2019.

[5] L. Lovén, T. Leppänen, E. Peltonen, J. Partala, E. Harjula, P. Porambage, M. Ylianttila, and J. Riekki, “EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks,” in *The 1st 6G Wireless Summit*, Levi, Finland, 2019, pp. 1–2.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.



(a) Small deployment (5 + 15 servers).



(b) Large deployment (15 + 5 servers).

Fig. 7: Comparison of the Small and Large deployment scenarios to the Reference scenario. The red boxes highlight a difference between both scaling scenarios and the reference, leading to spatial overlap in allocation.

[7] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 94–100, 2017.

[8] T. Leppänen *et al.*, "Developing Agent-Based Smart Objects for IoT Edge Computing: Mobile Crowdsensing Use Case," in *Internet and Distributed Computing Systems. IDCS 2018. Lecture Notes in Computer Science, vol 11226*, Y. Xiang, J. Sun, G. Fortino, A. Guerrieri, and J. J. Jung, Eds. Springer, Cham, 2018, pp. 235–247.

[9] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, 2009.

[11] F. Bonomi, R. Milito, J. Zhu, and 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.

[12] J. Haavisto, M. Arif, L. Lovén, T. Leppänen, and J. Riekki, "Open-source rans in practice: an over-the-air deployment for 5g mec," in *European Conference on Networks and Communications*, 2019, [To appear].

[13] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[14] T. Lähderanta *et al.*, "Edge server placement with capacitated location allocation," *arXiv preprint*, 2019. [Online]. Available: https://arxiv.org/pdf/1907.07349.pdf

[15] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, 2018.

[16] J. Liu, U. Paul, S. Troia, O. Falowo, and G. Maier, "K-means based spatial base station clustering for facility location problem in 5G," in *Proceedings of Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, J. Lewis and Z. Ndlela, Eds., Sept. 2018, pp. 406–409.

[17] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.

[18] H. Sinky, B. Khalfi, B. Hamdaoui, and A. Rayes, "Adaptive edge-centric cloud content placement for responsive smart cities," *IEEE Network*, vol. 33, no. 3, pp. 177–183, 2019.

[19] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1031–1045, 2017.

[20] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Software: Practice and Experience*, pp. 1–14, 2019.

[21] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*, ser. Contributions to Management Science. Physica-Verlag Heidelberg, 2009.

[22] J. Brimberg, P. Hansen, N. Mladenovic, and S. Salhi, "A survey of solution methods for the continuous location-allocation problem," *International Journal of Operations Research*, vol. 5, pp. 1–12, 01 2008.

[23] L. Cooper, "Heuristic methods for location-allocation problems," *SIAM Review*, vol. 6, no. 1, pp. 37–53, 1964. [Online]. Available: http://www.jstor.org/stable/2027512

[24] S. Tafazzoli and M. Marzieh, *Classification of Location Models and Location Softwares*, ser. Contributions to Management Science. Physica, Heidelberg, 07 2009, pp. 505–521.

[25] M. Negreiros and A. Palhano, "The capacitated centred clustering problem," *Computers & Operations Research*, vol. 33, no. 6, pp. 1639–1663, 2006.

[26] L. Kaufman and P. Rousseeuw, *Clustering by Means of Medoids*, ser. Delft University of Technology : reports of the Faculty of Technical Mathematics and Informatics. Faculty of Mathematics and Informatics, 1987.

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

[28] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in K-means clustering," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, no. 6, pp. 90–95, 2013.

[29] V. Kostakos, T. Ojala, and T. Juntunen, "Traffic in the smart city: Exploring city-wide sensing for traffic control center augmentation," *IEEE Internet Computing*, vol. 17, no. 6, pp. 22–29, Nov 2013.