A Learning-based Framework for Optimizing Service Migration in Mobile Edge Clouds

Florian Brandherm
TU Darmstadt
Darmstadt, Germany
brandherm@tk.tu-darmstadt.de

Lin Wang
VU Amsterdam
Amsterdam, The Netherlands
TU Darmstadt
Darmstadt, Germany
l.wang@vu.nl

Max Mühlhäuser
TU Darmstadt
Darmstadt, Germany
max@tk.tu-darmstadt.de

Abstract

Mobile edge computing is gaining traction due to its ability to deliver ultra-low-latency services for mobile applications. This is achieved through a federation of edge clouds in close proximity of users. However, the intrinsic mobility of users brings a high level of dynamics to the edge environment, calling for sophisticated service migration management across the edge clouds. Previous solutions for edge service placement/migration are architecture-specific, centralized, or are based on restricted cost models. These limitations leave doubts about the practicality of these approaches due to the lack of a standardized reference model for edge clouds. In this paper, we propose a general framework for optimizing edge service migration based on reinforcement learning techniques. Using our framework, edge service migration strategies can be learned with respect to a large variety of optimization goals. Moreover, our learning-based algorithm is agnostic to the underlying architecture and resource constraints. Preliminary results show that our model-free learning-based approach can compete with model-based baselines and adapt to different objectives.

CCS Concepts • **Networks → In-network processing**; *Cloud computing*; **In-network processing**; *Cloud computing*; • **Computing methodologies** → *Reinforcement learning*; *Reinforcement learning*;

ACM Reference Format:
Florian Brandherm, Lin Wang, and Max Mühlhäuser. 2019. A Learning-based Framework for Optimizing Service Migration in Mobile Edge Clouds. In *2nd International Workshop on Edge Systems, Analytics and Networking (EdgeSys '19), March 25, 2019, Dresden, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301418.3313939

1 Introduction

Mobile cloud computing enables applications on mobile devices that exceed their capabilities in terms of computation and storage. It also allows mobile devices to save energy by offloading energy-intensive computational tasks. Typically, mobile cloud computing is implemented through centralized data centers [25]. Despite their cost-effectiveness, those centralized data centers are usually far away from the end users, which denies the benefits from mobile cloud computing. It prohibits mobile applications that need to transfer large amounts of data or are subject to stringent latency requirements [25, 21, 11]. Examples for such applications are gaming-as-a-service [8], video streaming/processing [18], virtual/augmented reality [4] or autonomous driving [30].

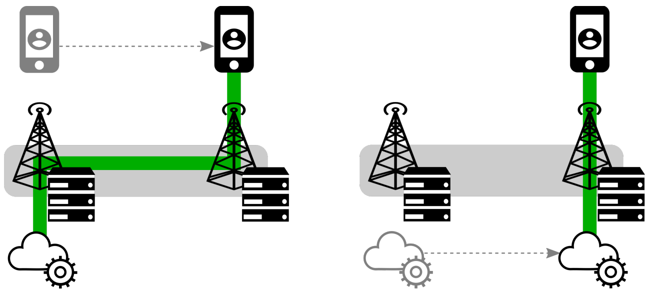

Figure 1. Service migration problem in a typical mobile edge cloud environment.

Locating cloud-infrastructure closer to the user equipment has the potential to reduce latency and backhaul traffic to a degree that enables the aforementioned applications. Following this idea, prominent architectures have been proposed under the names of mobile edge computing [1], cloudlets [19], or fog computing [16, 6]. In contrast to centralized cloud architectures, user mobility in mobile edge computing poses new challenges to resource management. More specifically, to ensure service quality, the service placement needs to be updated dynamically by service migrations to accommodate for changing user locations.

The present article focuses on the placement and migration of stateful services to deliver reliable low-latency- or high-throughput-services to mobile users. An example of the migration problem is given in Figure 1. Any user movement that results in an increased distance to its services degrades the latency and increases transmission costs for the network operator. Optimal service placement and migration are hard problems since the cost and latency impact of migrations and transmissions must be accounted for while avoiding congestion of limited resources in mobile edge clouds. While some early attempts have been made recently, existing proposals are typically restricted to specific architectures or over-simplified cost models—mainly due to tractability considerations—and thus are not generally applicable. In contrast, a data-driven, architecture-agnostic approach has the potential to be trained and deployed rapidly. Changes in the underlying architecture or cost model can be adopted without the need to develop new algorithms. In addition, the majority of previous approaches are centralized, also raising practicality and scalability concerns.

To overcome these limitations, we adopt an alternative approach, where we do not make any assumptions on the underlying architecture and cost model. In particular, we make the following contributions in this paper:

1. We present a general framework that can produce efficient service migration policies that can accommodate a large variety of objectives, and is largely independent of system architectures.
2. We map the adaptive service migration problem in distributed mobile edge clouds to a multi-agent reinforcement learning task and show that a policy can be learned using a singe-agent technique.
3. We carry out a preliminary evaluation and demonstrate that comparable performance to baseline approaches can be achieved by the learned policy, despite the fact that our policy can only observe local features. In contrast, the baselines use a model of the entire network and cost function.

The paper is organized as follows. Section 2 summarizes recent developments in edge service/placement and machine learning in networks. Section 3 introduces our system model and Section 4 describes the policy learning. Section 5 presents preliminary evaluation results. Section 6 discusses the limitations and future work. Section 7 concludes the paper.

2 Related Work

2.1 Edge Service Placement/Migration

As a specific challenge in mobile edge computing, service placement and migration has been widely studied in recent years [28, 23, 29, 26, 7, 27, 17]. The model by Urgaonkar et al. considers stateless services and includes heterogeneous costs [23]. It provides a strategy for request scheduling for multiple applications, given limited service rates, as well as the possibility to route requests to a backend cloud. The problem of stateful service migration is modeled by S. Wang et al. as an optimal decision problem that determines if a service should be migrated to its user location, exploiting the regular, hexagonal placement of base stations. The goal is to minimize the combined distance-based transmission and migration costs [28]. Ouyang et al. present two approximation algorithms to minimize total latency given a migration cost budget [17]. A cost prediction function is exploited by S. Wang et al. to search for cost-minimizing service placement sequences [29]. However, the authors don't evaluate their method with practical prediction methods. Resource allocation for mobile edge clouds is comprehensively studied by L. Wang et al. where service placement and migration are considered under specific cost models and an efficient online resource allocation algorithm is proposed, being agnostic to the mobility pattern of users [26]. He et al. explore the joint service placement and request scheduling under both sharable and unsharable resources [7]. Service migration is however omitted from their model. L. Wang et al. study the service placement problem specifically for virtual reality applications where they assume a polynomial performance interference model for service colocation. Service re-placement with user mobility is covered only by a simple heuristic and migration costs are not included in the optimization [27]. All of the above works suffer from one or more of the following limitations: being centralized, being specific to certain architectures, or assuming simplifying cost models.

2.2 Machine Learning for Networks

Machine learning techniques have been successfully applied in many of the control or optimization problems in the networking field, including cluster scheduling [14, 5], adaptive video streaming [13], congestion control [9], and traffic engineering [24, 3]. Mao et al. present DeepRM, an adaptive multi-resource cluster scheduling algorithm where a visual representation of cluster resources and jobs is fed to a deep reinforcement learning model [14]. More recently, Mao et al. introduced Decima, a reinforcement-learning-based model to schedule jobs that consist of a DAG containing a series of dependent tasks. Liu et al. present an approach for resource allocation and power management in cloud computing where the scheduling problem is divided into a hierarchy of 2 reinforcement learning problems. While the above scheduling and placement problems share some common aspects with the edge-cloud migration problem, they often expect an estimate of the job durations, and usually have no requirement to migrate tasks after they are placed. In contrast, stateful edge services may last as long as they are required and are affected by user mobility.

Reinforcement learning has also been applied to some traditional networking problems such as adaptive bitrate for video streams, congestion control, and traffic engineering. Mao et al. show that deep reinforcement learning can be used to improve the quality of bitrate control in adaptive video streaming [13]. Jay et al. make a preliminary attempt on how to better control the sending rate of TCP flows to achieve higher throughput or lower latency [9]. In [24] the routing decision is made by a reinforcement learning agent instead of a routing protocol. In addition to routing, Chen et al. consider the flow scheduling in data centers and applied a hybrid approach based on reinforcement learning and local control heuristics [3].

To the best of our knowledge, we are the first to use reinforcement learning techniques to provide a general approach for the edge service migration problem.

3 System Model

We consider the problem of service migration across resource-limited mobile edge clouds as a reaction to user movements. Although our algorithm is independent of the model specifications, the input state and possible actions are determined by the model. We present a general mobile edge cloud model with minimal assumptions to facilitate a model-free learning approach.

Wireless network model. Assume we are given a finite set of access points $\mathcal{B}$ and $l_b \in \mathbb{R}^2$ is the location of access point $b \in \mathcal{B}$. An access point can be any wireless gateway, such as a Wifi router or a cellular base station.

User mobility model. Denote by $b_u^t \in \mathcal{B}$ the access point that user u is connected to at time t. We make no further assumptions about the user mobility model, except that user speed is bounded.

Service model. Denote by $\Sigma_t \subset \Sigma$ the set of services that are present in the system at time t. Let $s_u \in \Sigma_t$ denote a service from user u at time t. The placement $\psi^t(s_u) : \Sigma_t \mapsto \mathcal{K}$ identifies the mobile edge cloud to which service s_u is assigned at time t.

Mobile edge cloud model. Denote the finite set of mobile edge clouds by $\mathcal{K}$. Each edge cloud cloud $\kappa \in \mathcal{K}$ has a dedicated memory resource with limited capacity $\widetilde{r}_\kappa^{\text{mem}}$ and current utilization $r_\kappa^{\text{mem}}(t)$. We make no assumption how $r_\kappa^{\text{mem}}(t)$ is calculated. There exists a hard limit on memory utilization $\forall t, 0 \leq r_\kappa^{\text{mem}}(t) \leq \widetilde{r}_\kappa^{\text{mem}}$, which implies that services cannot be executed if it would cause the memory utilization to exceed the available memory $\widetilde{r}_\kappa^{\text{mem}}$. If a new service is instantiated at an edge cloud with insufficient free resources, it remains inactive until it is migrated to an edge cloud with sufficient resources and subsequently activated. An already active service cannot be migrated to an edge cloud with insufficient resources. Instead, such migration attempts are rejected. A service is currently active if it is part of the set $\Sigma_t^{\text{active}} \subseteq \Sigma_t$. The cost of an inactive service $s_u \in \Sigma_t \setminus \Sigma_t^{\text{active}}$ is highly dependent on the specific application and cost model. While the cost can be very high for latency-bound applications, throughput-bound applications might be more tolerant. One might also imagine applications that utilize unused capacities, where the inactivity penalty could be very low.

4 Learning Service Migration Policies

The architecture and cost model of a mobile edge cloud system are often non-stationary and likely to change after deployment. For example, an edge-cloud provider may want to change its pricing model or extend its network with a new type of resource. To account for such flexibility, we propose a data-driven approach to quickly adapt the migration strategy without the need to manually re-engineer a new optimal migration algorithm.

Global RL problem. To adapt to different architectures and cost models, we propose to exploit reinforcement learning (RL) [20] to learn a cost-minimizing service migration strategy through trial and error. The service migration problem can be formulated as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. A stochastic policy $\pi_\theta(a|s)$ chooses an action $a \in \mathcal{A}$ based on the current state $s \in \mathcal{S}$. A stochastic transition $\mathcal{T}(s'|s, a)$ determines the following state s', given only the current state s and action a. The immediate value of choosing action a as a response to state s is measured by a stochastic reward function $\mathcal{R}(r|s, a)$. The goal in RL is to obtain a policy that maximizes the long-term expected discounted reward $\mathbb{E}_j\left[\sum_{i=0}^{\infty} \gamma^i r_{j+i}\right]$, where $0 \leq \gamma \leq 1$ is a discount factor that determines the influence of future rewards.

In a global mobile edge cloud model, the state s consists of the entire observable information at the time of a migration decision, which includes the state of all services and edge clouds in the system. An action is a migration $a_{s_u}^\kappa \in \mathcal{A}$, which means a service $s_u \in \Sigma_{t(i)}$ should be migrated to edge cloud κ. The transition function $\mathcal{T}(s'|s, a)$ is implemented by the present edge cloud architecture. The reward $r = \mathcal{R}(r|s, a) = -C(s, a)$ is the negative cost and depends on the current services and edge clouds.

Learning a policy for this global service migration problem is not scalable to large numbers of edge clouds and services. The state-action-space is the set of all possible placement transitions of all possible placements of all possible services. Known as the curse of dimensionality, this exponential growth quickly renders learning a global solution intractable. A practical limitation is that every decision and every reward computation would require to gather the entire system state, requiring large amounts of bandwidth and time.

Local RL problem. To alleviate these issues, we propose to learn an approximate solution by only considering the local region of each migration decision. This is motivated by the following observations:

1. migrations are usually only sensible over short distances because the speed of the users is limited
2. placement decisions in far away regions are of little consequence to each other.

We reformulate the problem as a competitive multi-agent learning problem, where each service aims to optimize its reward by deciding where to migrate. Therefore, the reward must be formulated for a single service, based on its observable region $\mathcal{N}_\kappa \subseteq \mathcal{K}$, which is predefined (e.g. K-nearest neighbors).

As local migration decisions are only based on a limited region around the service in question, the state-space is greatly reduced. The state consists only of the state of the $\|\mathcal{N}_\kappa\|$ clouds in the observable region. Similarly, the action space is greatly reduced by only allowing migrations to edge clouds within the observable region. Our service-centric formulation further removes the decision which services to migrate. Thus, the action space merely consists of the $\|\mathcal{N}_\kappa\|$ possible target clouds in the observable region plus a no-migration action. Moreover, since only local information is needed, individual decisions can be made locally and much faster than in a centralized system, even if part of the network fails.

Whenever a service is created or migrated, a migration decision is triggered. Additionally, a migration decision is triggered if the time since the last migration decision exceeds a certain threshold. When a migration decision occurs, first, the state of the local edge cloud neighborhood $\mathcal{N}_\kappa$ is gathered. Then, the policy $\pi_\theta(a|s)$ decides the optimal action, which can either be a migration to one of the edge clouds in $\mathcal{N}_\kappa$ or a no-migration action. Afterwards, an immediate reward for this service is computed.

Learning method. While the task is formulated as a multi-agent problem, such tasks can also be learned using single-agent algorithms [22]. Although they loose most of their theoretical guarantees in a multi-agent setting, they are often applied in practice [2]. We apply deep Q-learning with a target network [15]. It employs a deep function approximator to learn the state-action value $Q(s, a)$ that predicts the expected long-term reward of taking action a in state s. An ϵ-greedy policy (we chose $\epsilon = 0.1$ for our experiments)

$$\pi_\theta(a|s) = \begin{cases} \max_a Q(s, a) & \text{with} \quad p = 1 - \epsilon \\ a \sim U(a) & \text{with} \quad p = \epsilon \end{cases}$$

chooses mostly the exploitative action with the highest predicted value. Sometimes, it explores the action space by selecting a random action from the uniform distribution $U(a)$ over all possible actions. This randomization is necessary to collect new experiences for learning. We use a target network as described by Mnih et al. [15] to to alleviate some of the the detrimental effects of the non-stationary learning problem that arises from the multi-agent setting. We employ a multi-task paradigm, where a single policy generalizes over all services, which means that the policy parameters θ are shared globally. This accelerates learning and leads to a more general policy since all services share their experiences.

The choice of deep Q-learning has the added advantage that it is capable of off-policy learning, which means it can learn by observing the actions of another algorithm.

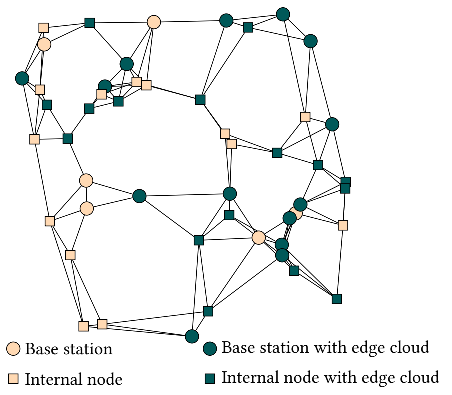

Figure 2. The randomly generated network architecture for mobile edge computing that we used in our evaluation.

5 Evaluation

Our simulated network architecture consists of 50 nodes (30 internal nodes and 20 base stations) which are randomly placed in a unit square. Each edge cloud is placed at a random node and can host at most 3 services. The nodes are first connected by a minimum spanning tree and additional connections are established between every node and its 4 nearest neighbors (euclidean distance). The resulting irregular network architecture is challenging for edge cloud migration algorithms as no regularities can be exploited. We depicted the network architecture we used in our evaluation in Figure 2. 40 users are simulated with a linear movement model, where the goal position is randomly selected within the unit square upon arrival at the previous goal. Every user is connected to its closest base station and is associated with one service. All users and services are created at the beginning of the simulation. Whenever a user connects to a new base station, the placement of the associated service is re-evaluated and a migration decision is triggered. A migration decision is also triggered, whenever the time since the last migration decision of a service exceeds a constant interval.

Every edge cloud has a local neighborhood $\mathcal{N}_\kappa$ containing the 10 nearest (topology-wise) edge clouds in the network. The observable state for the migration decision of a service s_u at edge cloud κ comprises the position, available memory $\widetilde{r}_\kappa^{\text{mem}}$ and currently used memory r_κ^{mem} of all edge clouds in $\kappa \cup \mathcal{N}_\kappa$, as well as the user's base station position, the previous cloud position, and the memory requirements of all services at the current edge cloud κ. While the state contains no aspects of the mobility model, a rough prediction can be learned implicitly from the base station and cloud positions. The state-action value function $Q(s, a)$ is approximated by a feed-forward neural network with 3 layers of 20 neurons. These parameters were determined by trial and error. For exploration, 10% of all actions are random ($\epsilon = 0.1$).

We evaluated our algorithm with a latency-minimization objective where the reward function $\mathcal{R}_1(s_u) = -\lambda(s_u)$ is the negative latency $\lambda(s_u)$, as determined by the number of hops between a service s_u and its user u.

5.1 Results

Latency-based reward function. We chose to evaluate our method with a latency-based objective because it is intuitive and good baselines can be implemented straightforwardly. Figure 3 depicts the average latencies and its variances of our method (blue) and compares it to two baseline strategies. The red line represents a never-migrate strategy. Drawn in black is a baseline strategy that greedily migrates services to the closest edge cloud in its observable neighborhood that has available memory resources. Note that this requires latency measurements which the learned strategy —being completely model-free— does not have. Despite the lack of direct latency measurements, our learned model is able to approach the performance of this baseline. This demonstrates that our method can learn good strategies without any knowledge about the cost model or edge cloud architecture. We also depict the performance of a global version of the always-migrate-strategy to show the loss in performance that is introduced by considering only a local view. However, this strategy disregards the communication overhead that is introduced by a centralized system that requires knowledge about the entire network.

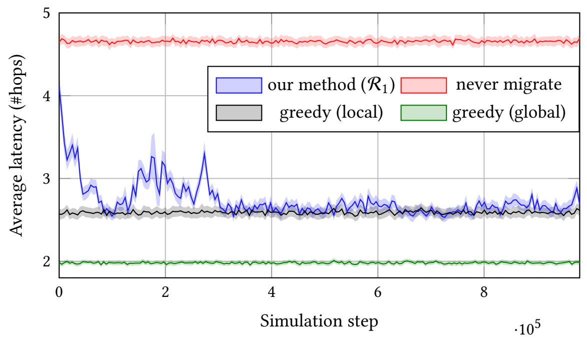

Figure 3. Latency comparison of $\mathcal{R}_1$.

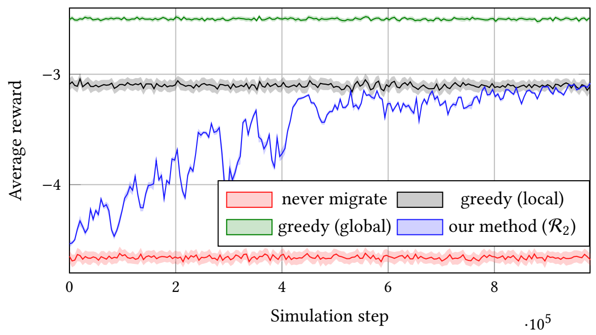

Figure 4. Reward comparison of $\mathcal{R}_2$.

Complex reward function. We also evaluated our system using another, more complex reward function to demonstrate that our algorithm can adapt to different cost models. The reward function $\mathcal{R}_2(s_u, \kappa) = -\lambda(s_u) - C_{\text{migration}}(s_u, \kappa)$ includes a migration cost $C_{\text{migration}}(s_u, \kappa)$ of 1 if s_u was migrated.

As depicted in Figure 4, which graphs the average rewards over time, the used learning method will improve upon the baseline performance of never migrating. The fact that our learning-based method can reach the performance of the hand-implemented baseline shows the potential of our learning-based method, considering that 10% of its actions are random. However, the performance during the learning process fluctuates considerably and convergence is comparatively slow. We see room for improvement by achieving better convergence properties and higher performance through a true multi-task formulation.

6 Discussion

Multi-agent problem. The service placement/migration problem in mobile edge clouds can be naturally treated as a multi-agent reinforcement learning problem since decisions have to be collaboratively made for all services in the mobile edge clouds. Currently, we apply a single-agent reinforcement learning approach, relying on the assumption that the mutual dependencies between services are low. While this method has already shown promising results, further investigations into learning techniques for multi-agent problems are expected and could lead to improved performance and better convergence.

Distributed learning. While the service migration policy is executed locally on each of the mobile edge clouds, enabling quick decisions, the training of the policy (for policy improvement) relies on centralized learning where the experience of all mobile edge clouds is gathered. A fully distributed approach would be to learn an individual policy locally on every mobile edge cloud, each specific to its local environment. One downside is that the generality can be lost. In other words, the learned policy of one mobile edge cloud cannot be applied to another with a different environment. This poses a problem for the dynamic addition of new edge clouds. To mitigate this issue, we plan to have further investigations into the domain of multi-task learning.

Local visibility. Another open question is the optimal selection of the local neighborhood $\mathcal{N}_\kappa$ that is considered visible to the mobile edge cloud. We see potential to reduce the performance gap between the global and local views by improving the local neighborhood selection. Our current implementation employs the K-nearest neighbors for each mobile edge cloud. However, we observe a tendency towards a clustering of nearby groups of mobile edge clouds, which leads to unbalanced clusters. We believe that this could be improved with more balanced, or even adaptive methods.

More comprehensive evaluation. Further efforts are required for a thorough evaluation with realistic user mobility patterns and different cost models. To understand the loss of our distributed approach, we will conduct a performance comparison between different neighborhood sizes, including the global view $\mathcal{N}_\kappa = \mathcal{K}$. To verify the generality of the proposed framework, we will carry out tests with different cost models and more heterogeneous system settings. Also, as indicated by Zheng et al. [31], it is necessary to examine the learned strategies to gain new insights and understand when the leaned model is likely to fail.

7 Conclusion

We targeted the adaptive service placement/migration problem in mobile edge clouds and presented a general framework for producing service migration strategies based on reinforcement learning techniques. Unlike existing approaches, the framework can generate efficient strategies without any knowledge or assumptions about the edge cloud architecture or cost model. Our preliminary results show that this approach already has the potential to achieve similar performance as the model-based baseline strategies. Being a first attempt, the work can be further refined following the discussion in the previous section.

8 Acknowledgement

This work has been funded by the German Research Foundation (DFG) as part of the projects A1 and C7 within the Collaborative Research Center (CRC) 1053—MAKI.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018.

[2] L. Bu, R. Babu, B. De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[3] L. Chen, J. Lingys, K. Chen, and F. Liu. Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *SIGCOMM*, pages 191–205, 2018.

[4] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler. Toward low-latency and ultra-reliable virtual reality. *IEEE Network*, 32(2):78–84, 2018.

[5] Y. Gao, L. Chen, and B. Li. Spotlight: optimizing device placement for training deep neural networks. In *ICML*, pages 1662–1670, 2018.

[6] J. Gedeon, J. Heuschkel, L. Wang, and M. Mühlhäuser. Fog computing: current research and future challenges. In *1. GI/ITG KuVS Fachgespräche Fog Computing*, 2018.

[7] T. He, H. Khamfroush, S. Wang, T. L. Porta, and S. Stein. It's hard to share: joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. In *ICDCS*, pages 365–375, 2018.

[8] C. Y. Huang, C. H. Hsu, Y. C. Chang, and K. T. Chen. Gaminganywhere: an open cloud gaming system. In *MMSys*, pages 36–47. ACM, 2013.

[9] N. Jay, N. H. Rotman, P. B. Godfrey, M. Schapira, and A. Tamar. Internet congestion control via deep reinforcement learning. In *Proceedings of the Deep Reinforcement Learning Workshop NeurIPS 2018*, pages 1–10, 2018.

[10] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *ICDCS*, pages 372–382. IEEE, 2017.

[11] P. Mach and Z. Becvar. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 2017.

[12] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh. Learning scheduling algorithms for data processing clusters. *arXiv preprint arXiv:1810.01963*, 2018.

[13] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *SIGCOMM*, pages 197–210, 2017.

[14] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *HotNets*, pages 50–56, 2016.

[15] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[16] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. A comprehensive survey on fog computing: state-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials*, 20(1):416–464, 2018.

[17] T. Ouyang, Z. Zhou, and X. Chen. Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications*, 2018.

[18] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos. Edge analytics in the internet of things. *IEEE Pervasive Computing*, (2):24–31, 2015.

[19] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[20] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[21] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. On multi-access edge computing: a survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys and Tutorials*, 19(3):1657–1681, 2017.

[22] M. Tan. Multi-agent reinforcement learning: independent vs. cooperative agents. In *ICML*, pages 330–337, 1993.

[23] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung. Dynamic service migration and workload scheduling in edge-clouds. *Performance Evaluation*, 91:205–228, 2015.

[24] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar. Learning to route. In *HotNets*, pages 185–191, 2017.

[25] B. Varghese and R. Buyya. Next generation cloud computing: new trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018.

[26] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser. Online resource allocation for arbitrary user mobility in distributed edge clouds. In *ICDCS*, pages 1281–1290, 2017.

[27] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser. Service entity placement for social virtual reality applications in edge computing. In *INFOCOM*, pages 468–476, 2018.

[28] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung. Dynamic service migration in mobile edge-clouds. In *IFIP Networking*, pages 1–9. IEEE, 2015.

[29] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung. Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1002–1016, 2017.

[30] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang. Toward cloud-based vehicular networks with efficient resource management. *IEEE Network*, 27(5):48–55, 2013.

[31] Y. Zheng, Z. Liu, X. You, Y. Xu, and J. Jiang. Demystifying deep learning in networking. In *Proceedings of the 2nd Asia-Pacific Workshop on Networking*, pages 1–7. ACM, 2018.