MOERA: Mobility-Agnostic Online Resource Allocation for Edge Computing

Lin Wang, Lei Jiao, Jun Li, Julien Gedeon, and Max Mühlhäuser

Abstract—To better support emerging interactive mobile applications such as those VR-/AR-based, cloud computing is quickly evolving into a new computing paradigm called edge computing. Edge computing has the promise of bringing cloud resources to the network edge to augment the capability of mobile devices in close proximity to the user. One big challenge in edge computing is the efficient allocation and adaptation of edge resources in the presence of high dynamics imposed by user mobility. This paper provides a formal study of this problem. By characterizing a variety of static and dynamic performance measures with a comprehensive cost model, we formulate the online edge resource allocation problem with a mixed nonlinear optimization problem. We propose MOERA, a mobility-agnostic online algorithm based on the "regularization" technique, which can be used to decompose the problem into separate subproblems with regularized objective functions and solve them using convex programming. Through rigorous analysis we are able to prove that MOERA can guarantee a parameterized competitive ratio, without requiring any a priori knowledge on input. We carry out extensive experiments with various real-world data and show that MOERA can achieve an empirical competitive ratio of less than 1.2, reduces the total cost by $4\times$ compared to static approaches, and outperforms the online greedy one-shot solution by 70 percent. Moreover, we verify that even being future-agnostic, MOERA can achieve comparable performance to approaches with perfect partial future knowledge. We also discuss practical issues with respect to the implementation of our algorithm in real edge computing systems.

Index Terms—Edge computing, resource allocation, online optimization, competitive analysis

1 Introduction

Mobile applications have been serving as fundamental elements in our daily life, providing functionalities such as social networking, online commerce, as well as entertainments. However, a critical problem has been observed where mobile devices are being overwhelmed by sophisticated mobile applications. On the one hand, modern mobile applications, especially those VR-/AR-based, require tremendous data processing (e.g., for scene rendering, object tracking and recognition). On the other hand, mobile devices, due to the fact that they are designed mainly for portability and energy efficiency, are capacity constrained in terms of both computing and storage.

To mitigate this resource mismatch, researchers have proposed various cloud-based solutions [1]. By leveraging the abundant resources available in distant clouds, computation intensive tasks from mobile applications can be devolved. One of the key issues in cloud-based task offloading is the large latency (usually larger than 100 ms according to our statistics [2]) between the mobile device and the distant cloud due to the multi-hop structure of the Internet core. However, for interactive mobile applications such as those VR-/AR-based, the ideal delay is usually less than 10 ms [3], meaning that the access delay for distant clouds is usually one order of magnitude larger than the requirement. Many other examples can be found in the cyber physical systems context, where time-critical decisions have to be made for applications including remote control of robotics, industrial automation, and autonomous driving [4]. This largely restricts the practicality of cloud-based solutions when it comes to real-world deployment, despite that many proof-of-concept prototypes have been developed. In addition, cloud-based task offloading requires to stream all the raw data to the distant cloud, resulting in substantial unnecessary traffic in the network.

Recently, edge computing was proposed to address these issues. Edge computing is a new paradigm that aims to bring computing or storage resources to the edge of the network. Connected by dedicated networks or the Internet, edge clouds may not have huge amounts of resources, but they are in close proximity to end users at various locations such as metropolitan centers, residential neighborhoods, cellular base stations, or even WiFi access points [5], [6], as illustrated in Fig. 1. Compared to distant clouds, serving end users from edge clouds has many advantages, which include lower or even bounded delay ($\sim$1 ms in 5G networks [7]) that can satisfy the requirement of advanced interactive mobile applications, reduced wide-area network traffic, and dedicated security or reliability.

One open challenge in the emerging edge computing paradigm is dynamic resource allocation for mobile applications that are running in edge clouds [8]. This task is nontrivial due to the fact that unlike large-scale distant clouds, edge clouds are more heterogeneous and dynamic. Many challenges are imposed by these unique features.

First, resource allocation for individual users may not be a single-cloud transaction in terms of both operation cost and service quality. When a user accesses the service in an edge cloud system, they may eventually have resources

• *L. Wang, J. Gedeon, and M. Mühlhäuser are with the Telecooperation Lab, Technische Universität Darmstadt, Darmstadt 64289, Germany. E-mail: {wang, gedeon, max}@tk.tu-darmstadt.de.*
• *L. Jiao and J. Li are with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403. E-mail: {jiao, lijun}@cs.uoregon.edu.*

Manuscript received 23 Aug. 2017; revised 2 July 2018; accepted 20 Aug. 2018. Date of publication 28 Aug. 2018; date of current version 28 June 2019. (Corresponding author: Lei Jiao.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2867520

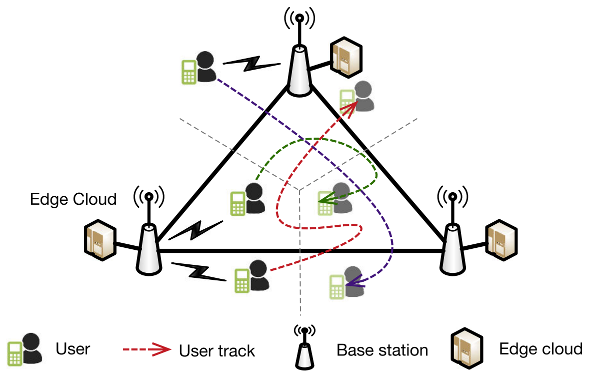


Fig. 1. An example scenario for edge computing. A set of Users are moving and are connected to different access points to access computing resources at the access points. While a user can appear in different physical locations over time, their service may not follow exactly the same trace due to the fact that frequent service migration is costly. On the other hand, serving a user from a different location will incur addition network latency which translates into service quality degradation.

allocated for them in multiple nearby edge clouds as a result of performance optimization performed by the edge cloud provider, as long as their service quality can be controlled. The user's perceived service quality in terms of the total access delay may include the network delay between the user and their access edge cloud they connect to and also that between their access edge cloud and all the other related edge clouds that hold their workload.

Further, resource allocation is not a one-shot task and needs to be continuously adapted to accommodate user movements, incurring the "adaptation cost" over time. Every user can move arbitrarily in the system, and, from a time-slotted view, a user may connect to the access point at one edge cloud in one time slot and switch to another in the next. In each time slot the system can have its own optimal resource allocation, which may, however, become suboptimal if the adaptation cost during time-slot transition is considered. The adaptation cost refers to hardware wear-and-tear (such as switching on/off a server) or the resource leading time (such as booting up or shutting down a virtual machine) [9], [10], [11]; it can also account for the bandwidth cost in the case of workload migration [12].

Finally, resource allocation needs to be performed on the fly, without any knowledge about future resource price and user location dynamics. It is usually hard or even impossible to predict how the resource price at each edge cloud will vary [10], [11] and how each user will move over time precisely [13]. Without such prediction, it is difficult to make an informed and good decision of resource allocation in each time slot; it is even more difficult to make decisions with guaranteed approximation towards the best decisions that can ever be made when assuming perfect knowledge about the future, which, however, is under our consideration.

Despite extensive existing research on resource allocation in the cloud context in general [14], [15], only a few have studied online resource allocation in edge clouds, falling short of addressing the three aforementioned challenges simultaneously. Most of the works often assume statistical knowledge about user mobility [16], [17], [18], or rely on prediction of future costs [19]. In addition, the resource adaptation cost has not been well considered until recently in the cloud in general [9] and in edge clouds in particular [10], [20]; nevertheless, none of them consider user mobility or its influence on resource allocation and adaptation.

1.1 Summary of Contributions

To the best of our knowledge, we are the first to present a formal study for optimizing the online resource allocation of edge clouds. We jointly consider the costs of allocation, reconfiguration, service quality, and migration in distributed edge clouds, under unpredictable resource prices and user movements. In particular, we make the following three contributions.

We build a comprehensive model to capture the optimization problem of online resource allocation in edge clouds. Our model includes four types of costs, but can capture a wide range of performance measures in general. We pursue the optimization of the total cost over time while serving user's workloads with the capacity limit of each edge cloud respected.

We transform our problem and propose an efficient online algorithm, for which, via rigorous competitive analysis, we prove a parameterized competitive ratio. Our major contribution is the design of an online algorithm based on the regularization technique [21], which decouples our original problem into a series of subproblems that are solvable in each independent time slot, only using the solution obtained for the previous time slot as input. The series of solutions generated in each time slot thus constitute a feasible solution to our original problem. By relaxation and primal-dual properties, we are able to demonstrate that our algorithm always outputs resource allocation decisions for each mobile user in each time slot, with a provable competitive guarantee even for the worst-case inputs.

We carry out extensive experiments to validate the performance of our proposed online algorithm. We use two real-world datasets for the evaluation. The results show that our algorithm produces near-optimal results regardless of the mobility pattern, with an empirical competitive ratio at most 1.2 in both real-world scenarios and outperforms the online greedy approach by up to 70 percent. We further test the algorithm with synthetic data and the results are consistent with those in the real-world cases, proving the effectiveness and generality of our algorithm. In addition, requiring zero future knowledge, our algorithm performs only slightly worse than approaches assuming perfect partial future knowledge. Finally, the remarkable gain of our algorithm is achieved only at the expense of moderate execution time, which still remains at the same level as in the online greedy approach.

1.2 Paper Organization

The remainder of this paper is structured as follows. Section 2 provides examples to motivate the work. Section 3 describes our models and formulates the problem. Section 4 focuses on the design details of the MOERA algorithm. Section 5 presents the formal competitive analysis. Section 6 describes the evaluations and interprets the results. Section 7 summarizes the related work. Section 8 concludes the paper and outlooks the future work.

2 Motivation

In this section, we motivate our work by explaining that online resource allocation should be made by taking into account multiple factors. In particular, we show that two major mobility-driven factors, namely system reconfiguration and workload migration, complicate the decision making for online resource allocation in such a dynamic edge computing environment.

System Reconfiguration. As no *a priori* information on future workload is assumed available, the simplest way would be to allocate the resource according to the current workload,

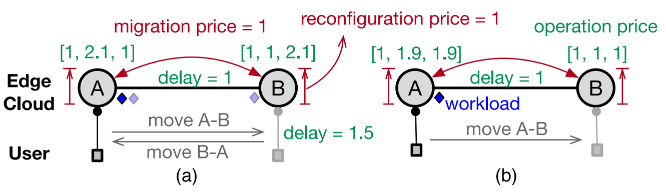


Fig. 2. Examples to show the complexity of decision making for online resource allocation for edge computing: (a) Shows that the greedy strategy can be too aggressive in the sense that migrating too often is harmful; (b) shows that the greedy strategy can be too conservative where migrations and migrations can be helpful if carefully done.

e.g., adding or removing resources continuously to follow workload changes. However, this can lead to very unexpected outcome where substantial cost has to be paid for system reconfiguration. It is usually assumed that removing resources from the system such as powering down a server or switching off a virtual machine can be negligible, but adding resources into the system can be costly. On the other hand, keeping idle resources active all the time would save this reconfiguration cost but will result in unnecessary operation costs. Therefore, a good decision making solution for resource allocation would be able to make intelligent tradeoffs between the two types of cost in order to achieve the optimal total cost and the real challenge consists in making such decisions on the fly, without knowing future workload.

Workload Migration. We provide two intuitive examples to show that workload migrations as a result of user mobility cannot be efficiently handled. We use the online greedy solution that achieves optimal resource allocation locally in every independent time slot for reference. We first consider the case in Fig. 2a, where we have a system with two edge clouds and one user is moving around them. The four types of prices, i.e., operation, service quality (measured by network delay), reconfiguration, and migration, are given in the figure and the user is assumed to have one unit of workload. We consider three time slots where in the first time slot the user is connected to edge cloud A and then, it moves to B in the second time slot and moves back to A in the third time slot. We now show that greedy can be too aggressive. Following the greedy approach, the user workload would be migrated from A to B in the second time slot. This is due to the fact that if the user workload stays at A, the incurred cost in the second time slot will be 4.6 (operation: 2.1, service-quality: 2.5, migration: 0, reconfiguration: 0), while the incurred total cost is 4.5 (operation: 1, service-quality: 1.5, migration: 1, reconfiguration: 1) if the user workload is migrated to B. The user workload would be migrated back to A again due to the fact that migrating to A would give a cost of 4.5 (operation: 1, service-quality: 1.5, migration: 1, reconfiguration: 1) while staying at B would give a cost of 4.6 (operation: 2.1, service-quality: 2.5, migration: 0, reconfiguration: 0) in the third time slot. The total cost for the three time slots would be calculated as $2.5 + 4.5 + 4.5 = 11.5$ in the greedy strategy, where both migration and reconfiguration costs are incurred in the last two time slots while the service quality cost is minimized as the workload is following the user all the time. However, with a holistic view on all the time slots, the optimal solution would keep the user workload at A throughout the three time slots, resulting in a total cost $2.5 + 4.6 + 2.5 = 9.6$ over the three time slots. The second example in Fig. 2b shows that the greedy solution can also be too conservative, where the greedy strategy would keep the workload all the time at A with a total cost of 11.3 (computed in the same way as in the previous example), while the optimal solution would migrate the workload to B in the second time slot and bring a total cost of only 9.5.

TABLE 1
List of Main Notations

Symbol	Meaning
T	set of time slots, i.e., $\{t_1, \ldots, t_h\}$
S	set of edge clouds, i.e., $\{s_1, \ldots, s_n\}$
C_s	capacity of edge cloud s
$d(s, s')$	delay between edge clouds s and s'
U	set of users, i.e., $\{u_1, \ldots, u_m\}$
λ_u	workload of user u
$l_{u,t}$	location of user u in time slot t
$s^*_{u,t}$	access edge cloud for user u in time slot t
$d(l_{u,t}, s^*_{u,t})$	access delay for user u in time slot t
$a_{s,t}$	operation price for edge cloud s in time slot t
c_s	reconfiguration price for edge cloud s
b^{out}_s	outbound migration price of edge cloud s
b^{in}_s	inbound migration price of edge cloud s
$w^{out}_{s,t}$	amount of workload being migrated out of edge cloud s in time slot t
$w^{in}_{s,t}$	amount of workload being migrated into edge cloud s in time slot t
$x_{s,u,t}$	amount of resources allocated to user u in edge cloud s in time slot t

Summary. The one-shot greedy solution is far from optimal in many cases and a holistic view is necessary. The situation becomes even worse when system reconfiguration is intwined with workload migration, meaning that online resource allocation decisions have to be made by jointly considering the two factors. Unfortunately, none of the existing algorithms would fit in this highly dynamic environment in a distributed edge computing system. Our motivation in this paper is thus to design an effective resource allocation algorithm that can simultaneously overcome all the above issues residing in the exiting solutions.

3 Problem Formulation

We present our models for the system, the user, and four types of cost, based on which we formulate the edge resource allocation problem in this section. Table 1 lists the main notations we will use throughout the paper.

3.1 Edge Cloud System

We consider a time-slotted system over h time slots, denoted by the set $T = \{t_1, \ldots, t_h\}$, where we assume that system settings will change across time slots and remain stable inside every time slot. We envisage an edge computing system with n edge clouds, denoted by $S = \{s_1, \ldots, s_n\}$, which are interconnected by a metropolitan area network (MAN). An edge cloud is a micro data center which is usually colocated with a cellular basestation or a WiFi access point. The hardware resources in the edge cloud are virtualized through some lightweight virtualization technology and thus, resources can be flexibly shared for multiplexing. Each edge cloud is equipped with a certain number of servers and the maximum capacity of an edge cloud $s \in S$ is given by C_s. The network delay between two edge clouds s_1 and s_2, i.e., the inter-cloud delay, is given by $d(s_1, s_2)$, where $d(s, s) \triangleq 0, \forall s \in S$. An edge cloud is supposed to cover a small geographical area and any user in the system will only receive coverage from the closest edge cloud.

3.2 User and Workload

We consider an edge-compatible mobile service where a set of m users, denoted by $U = \{u_1, \ldots, u_m\}$, are distributed in the considered metropolitan area and are moving around over time. In a certain time slot $t \in T$, a user $u \in U$ is assumed to be connected to the access point at an edge cloud $s^*_{u,t}$ that covers the vicinity of the user and offloads computation tasks to the edge clouds, incurring a workload of λ_u in total in the system. Taking augmented reality as an example, the computation tasks of a user mainly include object recognition and tracking as well as scene rendering. For the sake of tractability we only consider additive resources such as CPU and memory in our model.

Taking advantage of the heterogeneity of the edge clouds, the edge cloud operator may distribute the user workload to any of the edge clouds in order to achieve system-wide cost optimization; user workload may also be migrated across edge clouds over time in order to adapt to system dynamics. We denote by $x_{s,u,t}$ the amount of resources that are allocated for user u in edge cloud s at time t. We assume that only a subset of the edge clouds such as those with the closest proximity, denoted by $S_u \subseteq S$, is eligible for hosting the workload from user u. To accommodate the user workload from each user u successfully, we enforce the constraint that $\sum_{s \in S_u} x_{s,u,t} \geq \lambda_u$, meaning that the total amount of resources allocated for each user by the system should be no less than the workload of the user. The access delay for user u in time slot t, defined as the delay between the location $l_{u,t}$ of the user and her access point $s^*_{u,t}$,[1] is denoted by $d(l_{u,t}, s^*_{u,t})$. Standing as a major novelty of our model, no assumptions are made on user mobility patterns, i.e., $l_{u,t}$ can change arbitrarily over time.

3.3 Costs

The performance of the system is characterized with four general types of cost: the operation cost, the service quality cost, the reconfiguration cost, and the migration cost. The former two costs fall into the category of static cost that is independently incurred inside each time slot, while the latter two costs belong to the category of dynamic cost that is only charged for decision transitions across consecutive time slots.

Operation Cost. This cost refers to the usage of virtual machines including hardware resources such as CPU and memory, regular maintenance overhead on hardware or software, energy consumption, or even carbon emission, which is proportional to the total workload in each edge cloud. Denote by $a_{s,t} > 0$ the "operation price", i.e., the cost for each unit of workload, of edge cloud s in time slot t. The total operation cost in the edge computing system can be generally captured by

$$E_O = \sum_{t \in T} \sum_{s \in S} a_{s,t} \sum_{u \in U} x_{s,u,t}. \tag{1}$$

Note that we allow arbitrary variations on the operation price over time, and such variations can be heterogeneous for different edge clouds due to different hardware or software specifications or energy prices.

Service Quality Cost. This cost aims to capture the user-perceived quality of service, which is proportional to the network delay between the user and her workload. While the workload of a user may be distributed to multiple edge clouds for the sake of cost optimization, the user-perceived quality of service must be controlled. For a given edge cloud s and a user u, the service quality cost is characterized by the user's access delay $d(l_{u,t}, s^*_{u,t})$ and the weighted sum of the delay between the access edge cloud and each of the edge clouds that host the workload of user u. As a result, the total service quality cost in the system can be expressed by

$$E_Q = \sum_{t \in T} \sum_{u \in U} \left(d(l_{u,t}, s^*_{u,t}) + \sum_{s \in S_u} \frac{x_{s,u,t}}{\lambda_u} d(s^*_{u,t}, s) \right). \tag{2}$$

Reconfiguration Cost. This cost is associated with the increase of workload across time slots in each edge cloud. As users move, the edge cloud provider may redistribute the workload from each user to reduce the service quality cost, which results in adapting the amount of resources being allocated in each edge cloud. Such adaptation involves powering up physical servers, which would incur some inevitable delay due to hardware or software preparation and some implicit cost caused by frequent hardware wear-and-tear. We assume the reconfiguration cost is proportional to the amount of increased workload and the reconfiguration price, i.e., the cost for increasing unit resource, is given by $c_s > 0$ for each cloud $s \in S$. By defining function $(x)^+ = \max\{x, 0\}$ for all $x \in \mathbb{R}$, the total reconfiguration cost is calculated as

$$E_R = \sum_{t \in T} \sum_{s \in S} c_s \left(\sum_{u \in U} x_{s,u,t} - \sum_{u \in U} x_{s,u,t-1} \right)^+, \tag{3}$$

where $\left(\sum_{u \in U} x_{s,u,t} - \sum_{u \in U} x_{s,u,t-1}\right)^+$ captures the workload increase in edge cloud s when transitioning from time slot $(t-1)$ to time slot t. The cost associated with removing resources is omitted here as that can usually be completed without bringing extra delay to the user and thus, the cost is negligible.

Migration Cost. This cost characterizes the overhead incurred by migrating some workload from one edge cloud to another. This overhead includes the bandwidth cost on the network and the migration delay. We denote by b^{out}_s and b^{in}_s the migration price, i.e., the cost of migrating unit workload, associated with data moving out of and into edge cloud p, respectively and by $w^{out}_{s,t}$ and $w^{in}_{s,t}$ the amount of workload being migrated out of and into edge cloud s at time t, respectively. We have the following equations:

$$\begin{aligned} w^{out}_{s,t} &= \sum_{u \in U} (x_{s,u,t-1} - x_{s,u,t})^+, \\ w^{in}_{s,t} &= \sum_{u \in U} (x_{s,u,t} - x_{s,u,t-1})^+. \end{aligned} \tag{4}$$

The total migration cost E_Q thus can be captured by

$$E_M = \sum_{t \in T} \sum_{s \in S} b^{out}_s w^{out}_{s,t} + b^{in}_s w^{in}_{s,t}. \tag{5}$$

All the above cost models are illustrated in Fig. 3. We believe that these cost models are general enough and can capture a wide range of practical performance measures in an edge computing system from the perspective of edge cloud provider. Note that there is a big difference between the reconfiguration cost and the migration cost, i.e., the migration cost is calculated independently for each user,

1. With a slight abuse of notation, we also use the edge cloud $s^*_{u,t}$ to represent the access point that user u is connected to in time slot t.

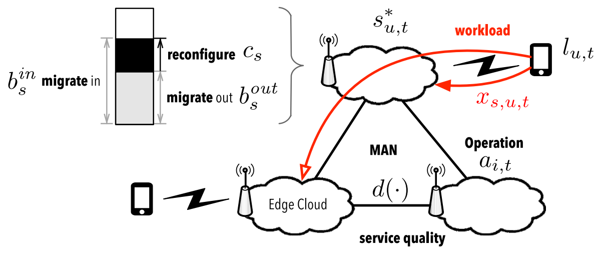


Fig. 3. The breakdown of the overall system cost: Operation cost, service quality cost, reconfiguration cost, and migration cost.

while the reconfiguration cost is associated with the collaborative workload changes on each edge cloud.

3.4 Problem Formulation

The overall performance measure, i.e., the total cost of the system, is defined as the weighted sum of all the aforementioned costs, as given by

$$E = E_O + E_R + E_Q + E_M. \tag{6}$$

For the simplicity of expression, we omit the tradeoff weights in our models. Nevertheless, these weights can be incorporated implicitly in the parameters in the cost models, e.g., $a_{s,t}$ for the operation cost and c_s for the reconfiguration cost. We will discuss the impact of the weights in our evaluations. Our goal is to develop an online algorithm which takes the user's workload and location as input and continuously decides how much resources to be allocated in each edge cloud that belongs to subset S_u, such that the workload demands from every user can be satisfied while the overall cost of the edge computing system is minimized over time. In each time slot $t \in T$, the resource allocation decision $x_{s,u,t-1}$ for the previous time slot $(t-1)$ might become suboptimal due to the variation of $a_{s,t}$ as well as the change of $l_{u,t}$ as a result of user movement. Therefore, the optimizer will need to redistribute the workload among all the edge clouds in order to maintain optimal cost efficiency. However, the redistribution of workload comes with additional (dynamic) costs, i.e., E_R for reconfiguring the edge clouds and E_M for migrating the workload. Ideally, the optimizer would make the best tradeoff between the static and the dynamic costs.

Combining all the aforementioned models, the edge resource allocation problem can further be formulated with the following linear program, denoted as $\mathbb{P}_0$

$$\begin{aligned} \min \quad & P_0 = \overbrace{E_O + E_Q}^{\text{static}} + \overbrace{E_R + E_M}^{\text{dynamic}} \\ \text{s.t.} \quad & \sum_{s \in S_u} x_{s,u,t} \geq \lambda_u, \ \forall u, \ \forall t, && (7a) \\ & \sum_{u \in U} x_{s,u,t} \leq C_s, \ \forall s, \ \forall t, && (7b) \\ & x_{s,u,t} \geq 0, \ \forall s, \ \forall u, \ \forall t. && (7c) \end{aligned}$$

Constraint (7a) ensures sufficient resources are allocated for every user; constraint (7b) guarantees that the capacity constraint for each edge cloud is not violated. Note that all the costs have time-varying factors corresponding to the uncertainties or dependencies across consecutive time slots. As a result, there is no once-for-all solution for the problem in each separate time slot from an online perspective.

We observe that the problem can be solved by directly applying a linear program solver (e.g., GLPK) if we are given in advance all the input data including the operation prices and the user mobility patterns in all time slots. However, this is impossible in the online setting, where the input data are revealed step by step over time. Without any a priori knowledge, a natural solution would be greedily adopting the best decision in each independent time slot. However, we have already shown (see Section 2) by concrete examples that the online greedy approach is suboptimal for multiple reasons. Thus, we aim to develop an efficient online algorithm that can deal with arbitrary system dynamics.

4 Online Algorithm Design

We present MOERA - *Mobility-agnostic Online Edge Resource Allocation* for the formulated problem. At the beginning, MOERA carries out a gap-preserving transformation to simplify the original problem. MOERA is then based on solving a subproblem with a carefully designed logarithmic objective in each time slot, and the solutions for all the subproblems will finally constitute a feasible solution for the original resource allocation problem over time.

4.1 Problem Transformation

We notice that the migration cost in $\mathbb{P}_0$ is counted bidirectionally, i.e., workload migration from edge cloud s_1 to edge cloud s_2 would incur both outbound migration cost at s_1 and inbound migration cost at s_2, which is too complicated to handle. To simplify this expression, we carry out a transformation on the migration cost in the objective of $\mathbb{P}_0$, from which we generate the following new linear program $\mathbb{P}_1$

$$\begin{aligned} \min \quad & P_1 = E_O + E_R + E_Q + \sum_{t \in T} \sum_{s \in S} b_s w_{s,t}^{in} \\ \text{s.t.} \quad & (7a), (7b), (7c), \end{aligned}$$

where we define $b_s \triangleq b_s^{out} + b_s^{in}$. The intuition behind this transformation is to combine the migration costs counted in both directions into a combined cost that is counted on only one direction. The transformation is gap-preserving while improving the tractability of the problem. More specifically, by following similar techniques used in [12], we are able to show that

Lemma 1. *Any r-competitive online algorithm that solves $\mathbb{P}_1$ also yields a r-competitive online algorithm for $\mathbb{P}_0$.*

Proof. For each edge cloud $s \in S$, we have that the accumulative workload x_s during the whole time period $[t_1, t_h]$ is bounded by the capacity C_s of the edge cloud, i.e.,

$$x_s = \left| \sum_{t \in T} w_{s,t}^{in} - \sum_{t \in T} w_{s,t}^{out} \right| \leq C_s. \tag{8}$$

Consequently, the following result can be derived

$$\begin{aligned} P_0 &= \sum_{t \in T} \sum_{s \in S} (b_s^{out} w_{s,t}^{out} + b_s^{in} w_{s,t}^{in}) \\ &= \sum_{t \in T} \sum_{s \in S} \left(b_s^{out} \left(w_{s,t}^{in} \pm x_s \right) + \sum_{s \in S} b_s^{in} w_{s,t}^{in} \right) \\ &\geq \sum_{t \in T} \sum_{s \in S} (b_s^{out} + b_s^{in}) w_{s,t}^{in} - \sum_{t \in T} \sum_{s \in S} b_s^{out} C_s \\ &\geq P_1 - \sum_{t \in T} \sum_{s \in S} b_s^{out} C_s. \end{aligned} \tag{9}$$

As $\sigma = \sum_{t\in T}\sum_{s\in S} b_s^{out} C_s$ is a constant and we have that $P_1 \le P_0 + \sigma$, this indicates that P_1 is upper bounded by P_0 plus a constant σ. This completes the proof as any online algorithm that produces a solution with objective value bounded by r times the optimal of $\mathbb{P}_1$ will also be a solution that is bounded by r times the optimal of $\mathbb{P}_0$ within a constant $r\sigma$. □

The above result provides us the convenience to consider only the problem $\mathbb{P}_1$ hereafter. We also observe that the migration cost can be decomposed individually for each of the users. By introducing auxiliary variables $w_{s,u,t}$ where we define $w_{s,u,t} = (x_{s,u,t} - x_{s,u,t-1})^+$ and combining with $w_{s,t}^{in} = \sum_{u\in U} w_{s,u,t}$, we rewrite the objective function of $\mathbb{P}_1$ as follows:

$$P_1 = E_O + E_R + E_Q + \sum_{t\in T}\sum_{s\in S}\sum_{u\in U} b_s w_{s,u,t}. \tag{10}$$

4.2 Design of MOERA

We now present the design of the proposed algorithm for the edge resource allocation problem. To measure the quality of the solutions produced by an online algorithm, we introduce *competitive ratio*, which is defined as the ratio of the objective of an online algorithm for a given online optimization problem where the input is revealed over time and the optimal objective obtained assuming all the input for the problem is pre-given, i.e., offline optimal. To simplify the presentation, we denote by $x_{s,t}$ the total amount of resources allocated in edge cloud s in time slot t, i.e., $x_{s,t} = \sum_{u\in U} x_{s,u,t}$.

The MOERA algorithm is based on the algorithmic technique called regularization [21], i.e., solving $\mathbb{P}_1$ with regularized objective functions. The pseudo code of MOERA is listed in Algorithm 1. At the beginning of each time slot $t \in T$, observing $l_{u,t}$ and taking $x^*_{s,u,t-1}$ ($x^*_{s,u,0} \triangleq 0$) which is the workload assignment decision made in time slot $(t-1)$, as inputs, we solve the following problem $\mathbb{P}_2(t)$ and obtain the resource allocation decisions $x^*_{s,u,t}$ for the current time slot t

$$\begin{aligned}
\min \quad & P_2(t) = \sum_{s\in S}\sum_{u\in U} a_{s,t} x_{s,u,t} \\
& + \sum_{u\in U}\left(d(l_{u,t}, s^*_{u,t}) + \sum_{s\in S_u} \frac{x_{s,u,t}}{\lambda_u} d(s^*_{u,t}, s)\right) \\
& + \sum_{s\in S}\frac{c_s}{\eta_s}\left((x_{s,t}+\varepsilon_1)\ln\frac{x_{s,t}+\varepsilon_1}{x^*_{s,t-1}+\varepsilon_1} - x_{s,t}\right) \\
& + \sum_{s\in S}\sum_{u\in U}\frac{b_s}{\tau_{s,u}}\left((x_{s,u,t}+\varepsilon_2)\ln\frac{x_{s,u,t}+\varepsilon_2}{x^*_{s,u,t-1}+\varepsilon_2} - x_{s,u,t}\right) \\
\text{s.t.} \quad & \sum_{s\in S_u} x_{s,u,t} \ge \lambda_u \ \forall u,
\end{aligned} \tag{11a}$$

$$\sum_{k\in S\setminus s}\sum_{u\in U} x_{k,u,t} \ge \sum_{u\in U}\lambda_u - C_s, \ \forall s, \tag{11b}$$

$$x_{s,u,t} \ge 0, \ \forall s, \ \forall u, \tag{11c}$$

where $\eta_s = \ln(1 + C_s/\varepsilon_1)$, $\tau_{s,u} = \ln(1+\lambda_u/\varepsilon_2)$, and $\varepsilon_1 > 0$, $\varepsilon_2 > 0$ are parameters. Note that the objective function $P_2(t)$ is convex and the constraints are all linear. As a result, $\mathbb{P}_2(t)$ can be optimally solved by any solver for convex programs. Combining the optimal solution to $\mathbb{P}_2(t)$ in every time slot $t\in T$, denoted by $x^*_{s,u,t}$, we construct an approximate solution to the original problem $\mathbb{P}_1$ by simply following exactly the same resource allocation decisions. We show in the following that the produced solution is feasible to $\mathbb{P}_1$ inherently.

Algorithm 1. MOERA

```
1: gap-preserving transformation $\mathbb{P}_0 \Rightarrow \mathbb{P}_1$;
2: regularize the objective $P_1 \Rightarrow P_2(t)$ for $t \in T$;
3: initialize $x_{s,u,0} \leftarrow 0$ for all $s \in S, u \in U$;
4: for $t \in T$ do                         ▷ online resource allocation
5:     update $P_2(t)$ using $l_{u,t}$ and $x^*_{s,u,t-1}$;
6:     solve $\mathbb{P}_2(t)$ via convex programming;
7:     $t \leftarrow t + 1$;
8: end for
```

Theorem 1 (Feasibility). *The optimal solution $x^*_{s,u,t}$ to $\mathbb{P}_2(t)$ in every time slot $t \in T$ constitutes a feasible solution to $\mathbb{P}_1$.*

Proof Sketch. The proof is conducted by showing that the optimal solution obtained for $\mathbb{P}_2(t)$ also satisfies the constraints (7a), (7b), and (7c) in $\mathbb{P}_1$. For more details please refer to Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2018.2867520. □

4.3 Time Complexity

It is convenient to check that the MOERA algorithm can be finished in polynomial time since it only relies on solving a series of convex programs over time, which are known to be polynomial time solvable [22]. There are many solution approaches available for convex programming, from which we choose the Interior Point method due to its practical performance. In general, the Interior Point method converges in time $O((mn)^{3.5})$ where mn is the total number of variables in the convex program instance $\mathbb{P}_2(t)$ in each time slot t.

5 Competitive Analysis

In this section, we carry out rigorous theoretical analysis on the performance of MOERA following the definition of competitive analysis. Competitive analysis is a standard method for quantifying the performance of online algorithms. The general idea is to compare the performance of an online algorithm to that of the optimal offline algorithm that can view the input sequence in advance. In particular, we prove that our algorithm has guaranteed performance by showing a parameterized competitive ratio. We first present the rationale behind the proof, followed by the detailed derivations step by step.

5.1 Rationale

The competitive analysis for MOERA will be conducted following three high-level steps:

(1) relax the program $\mathbb{P}_1$ by linearizing the objective function via introducing auxiliary variables $y_{s,t}$ and $z_{s,u,t}$, from which we obtain a linear program $\mathbb{P}_3$;
(2) derive the dual problem of $\mathbb{P}_3$ to obtain a program $\mathbb{D}$;
(3) construct a feasible solution for $\mathbb{D}$ from the optimal solution $x^*_{s,u,t}$ generated by optimally solving $\mathbb{P}_2(t)$.

The rationale of adopting the regularization-based approach is that the optimal solution for $\mathbb{P}_2(t)$ shares some common properties with the solution we constructed for $\mathbb{D}$, which can be exploited by deriving the Karush-Kuhn-Tucker (KKT) conditions, i.e., the first-order necessary conditions for a solution to be optimal, for $\mathbb{P}_2(t)$ and compare them with the constraints in $\mathbb{D}$. As a result, a connection between the optimal solution to $\mathbb{P}_2(t)$ and the constructed solution to $\mathbb{D}$ can be established. More formally, we aim to derive the following inequalities:

$$P_1 \geq P_3 \geq D \geq \frac{1}{r}\sum_{t\in T} P_2(t), \tag{12}$$

where the inequality $P_1 \geq P_3$ follows by the fact that $\mathbb{P}_3$ is relaxed from $\mathbb{P}_1$, as a result of which it produces the -optimal solution no larger than that of $\mathbb{P}_1$. The inequality $P_3 \geq D$ is obtained by applying the Weak Duality Theorem, and the inequality $rD \geq P_2$ follows by comparing the solutions to $\mathbb{D}$ with that to $\mathbb{P}_2(t)$. These inequalities together lead to the result that the proposed online algorithm MOERA, is r-competitive, where the competitive ratio r will be determined later.

5.2 Auxiliary Programs

Following the above rationale, we first provide the formulation for the relaxed program $\mathbb{P}_3$. As already mentioned, we introduce auxiliary variables $y_{s,t}$ and $z_{s,u,t}$ to reformulate the nonlinear terms in the objective function of $\mathbb{P}_1$. We also enforce lower bounds on the new variables in the constraints. The formal formulation of $\mathbb{P}_3$ is given below:

$$\begin{aligned}
\min \quad & P_3 = \sum_{t\in T}\sum_{s\in S}\sum_{u\in U} a_{s,t}x_{s,u,t} + \sum_{t\in T}\sum_{u\in U}\sum_{s\in S_u}\frac{x_{s,u,t}}{\lambda_u} d(s^*_{u,t}, s) \\
& \quad + \sum_{t\in T}\sum_{s\in S} c_s y_{s,t} + \sum_{t\in T}\sum_{s\in S}\sum_{u\in U} b_s z_{s,u,t} && (13a)\\
\text{s.t.} \quad & y_{s,t} \geq \sum_{u\in U} x_{s,u,t} - \sum_{u\in U} x_{s,u,t-1},\ \forall s,\ \forall t, \\
& z_{s,u,t} \geq x_{s,u,t} - x_{s,u,t-1},\ \forall s,\ \forall u,\ \forall t, && (13b)\\
& \sum_{k\in S\backslash s}\sum_{u\in U} x_{k,u,t} \geq \left(\sum_{u\in U}\lambda_u - C_s\right)^+,\ \forall s,\ \forall t, && (13c)\\
& y_{s,t} \geq 0,\ \forall s,\ \forall t, && (13d)\\
& z_{s,u,t} \geq 0,\ \forall s,\ \forall u,\ \forall t, \\
& (7a), (7c), && (13e)
\end{aligned}$$

where function $(x)^+$ can be applied to the right-hand term of (13c) due to the fact that $x_{s,u,t} \geq 0$. Note that we omit $\sum_{t\in T}\sum_{u\in U} d(l_{u,t}, s^*_{u,t})$ from the service quality cost E_Q because this component of cost is independent of the resource allocation decision once the user is connected to a base station. We now derive the Lagrangian dual of $\mathbb{P}_3$ to generate program $\mathbb{D}$. To this end, we introduce dual variables for each of the constraints in $\mathbb{P}_3$: Let $\alpha_{s,t}$, $\beta_{s,u,t}$, $\rho_{s,t}$, and $\theta_{u,t}$ be the dual variables associated with (13a), (13b), (13c), and (7a), respectively. Denote by g_s an indicator where $g_{s,u} = 1$ if $s \in S_u$ for $u \in U$ and $g_{s,u} = 0$ otherwise. The dual program $\mathbb{D}$ can be derived as follows:

$$\begin{aligned}
\max \quad & D = \sum_{t\in T}\sum_{u\in U}\lambda_u g_{s,u}\theta_{u,t} + \sum_{t\in T}\sum_{s\in S}\left(\sum_{u\in U}\lambda_u - C_s\right)^+ \rho_{s,t} && (14a)\\
\text{s.t.} \quad & -a_{s,t} - g_{s,u}\frac{d(s^*_{u,t}, s)}{\lambda_u} + \alpha_{s,t+1} - \alpha_{s,t} + \beta_{s,u,t+1} \\
& -\beta_{s,u,t} + \sum_{k\in S\backslash s}\rho_{k,t} + g_{s,u}\theta_{u,t} \leq 0,\ \forall s,\ \forall u,\ \forall t, && (14b)\\
& -c_s + \alpha_{s,t} \leq 0,\ \forall s,\ \forall t, && (14c)\\
& -b_s + \beta_{s,u,t} \leq 0,\ \forall s,\ \forall u,\ \forall t && (14d)\\
& \alpha_{s,t} \geq 0, \rho_{s,t} \geq 0\ \forall s,\ \forall t, && (14e)\\
& \beta_{s,u,t} \geq 0, \theta_{u,t} \geq 0,\ \forall s,\ \forall u,\ \forall t. && (14f)
\end{aligned}$$

On the other hand, we derive the KKT conditions of the program $\mathbb{P}_2(t)$. We associate dual variables $\theta'_{u,t}$, $\rho'_{s,t}$, and $\delta'_{s,u,t}$ to constraints (11a), (11b), and (11c), respectively. Consequently, we have

$$\begin{aligned}
& a_{s,t} + g_{s,u}\frac{d(s^*_{u,t}, s)}{\lambda_u} + \frac{c_s}{\eta_s}\ln\frac{x_{s,t}+\varepsilon_1}{x^*_{s,t-1}+\varepsilon_1} + \frac{b_s}{\tau_{s,u}}\ln\frac{x_{s,u,t}+\varepsilon_2}{x^*_{s,u,t-1}+\varepsilon_2} \\
& - g_{s,u}\theta'_{u,t} - \sum_{k\in S\backslash s}\rho'_{k,t} - \delta'_{s,u,t} = 0,\ \forall s,\ \forall u, && (15a)
\end{aligned}$$

$$\theta'_{u,t}\left(\lambda_u - \sum_{s\in S_u} x_{s,u,t}\right) = 0,\ \forall u, \tag{15b}$$

$$\rho'_{s,t}\left(\sum_{u\in U}\lambda_u - C_s - \sum_{k\in S\backslash s}\sum_{u\in U} x_{k,u,t}\right) = 0,\ \forall s \tag{15c}$$

$$-x_{s,u,t}\delta'_{s,u,t} = 0,\ \forall s,\ \forall u, \tag{15d}$$

$$(11a), (11b), (11c), \theta'_{u,t} \geq 0, \rho'_{s,t} \geq 0,\ \forall s,\ \forall u, \tag{15e}$$

where Equation (15a) is due to stationarity; Equations (15b), (15c), and (15d) are due to complementary slackness; inequalities in (15e) are due to primary or dual feasibility. Using the optimal solutions obtained from solving $\mathbb{P}_3$ in time slot t, i.e., $x^*_{s,u,t}$ and the dual variables $\theta'_{u,t}$ and $\rho'_{s,t}$, we construct a solution S_D for program $\mathbb{D}$ by following the mappings below:

$$\begin{aligned}
& \alpha_{s,t} = \frac{c_s}{\eta_s}\ln\frac{C_s+\varepsilon_1}{x^*_{s,t-1}+\varepsilon_1}, \beta_{s,u,t} = \frac{b_s}{\tau_{s,u}}\ln\frac{C_s+\varepsilon_2}{x^*_{s,u,t-1}+\varepsilon_2} \\
& \theta_{u,t} = \theta'_{u,t}, \rho_{s,t} = \rho'_{s,t}.
\end{aligned}$$

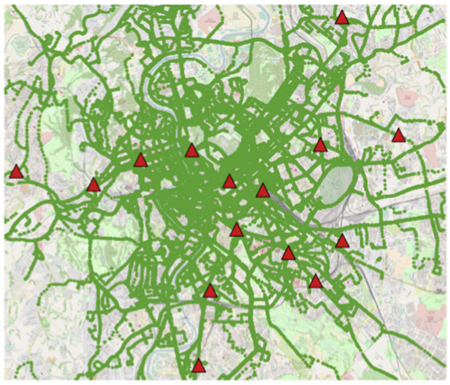
We are able to show that

Lemma 2. *The solution S_D is feasible for program $\mathbb{D}$.*

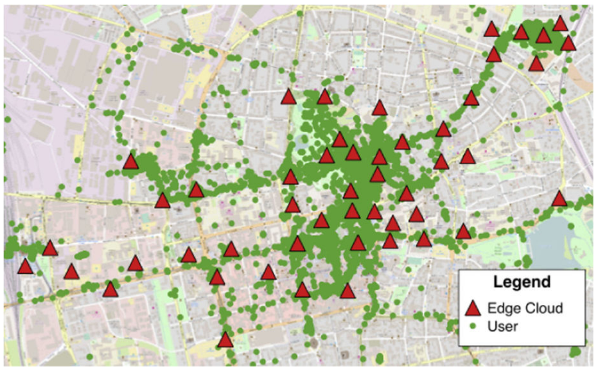
Proof Sketch. The proof is conducted by showing that when substituted in $\mathbb{D}$, S_D satisfies all the constraints. For details please refer to Appendix B, available in the online supplemental material. □

5.3 Competitive Ratio

We now focus on the last inequality $rD \geq \sum_{t\in T} P_2(t)$ in (12) and derive the competitive ratio r. We first partition the total cost into two parts and show that each part can be bounded by a certain factor. Then, we combine the results for both parts and derive the competitive ratio.



(a) Rome



(b) Darmstadt

Fig. 4. Distributions of edge clouds and users in the two selected cities: Rome (Italy) and Darmstadt (Germany).

Lemma 3. *The total static cost, i.e., the sum of the operation cost and the service quality cost in* $\mathbb{P}_1$, *is upper bounded by* D *when evaluated at* $x^*_{s,u,t}$, *i.e.,* $E_O + E_Q \leq D$.

Proof Sketch. The proof is conducted by applying the Equations (15a), (15b), (15c), and (15d) obtained from the KKT conditions of $\mathbb{P}_2(t)$ to the total static cost in the objective function $P_2(t)$. For more details please refer to Appendix C, available in the online supplemental material. □

Lemma 4. *The total dynamic cost, i.e., the sum of the reconfiguration cost and the migration cost in* $\mathbb{P}_1$, *is upper bounded by constant times of* D *when evaluated at* $x^*_{s,u,t}$, *i.e.,* $E_R + E_M \leq \gamma n D$ *where*

$$\gamma = \max_{s\in S}\left\{(C_s+\varepsilon_1)\ln\left(1+\frac{C_s}{\varepsilon_1}\right), (C_s+\varepsilon_2)\ln\left(1+\frac{C_s}{\varepsilon_2}\right)\right\}.$$

Proof Sketch. The proof is conducted by applying the Equations (15a), (15b), (15c), and (15d) obtained from the KKT conditions of $\mathbb{P}_2(t)$ to the total dynamic cost in the objective function $P_2(t)$. For more details please refer to Appendix D, available in the online supplemental material. □

Combining all the results in Theorem 1, Lemmas 3, and 4, the following theorem on the competitive ratio can be obtained for our proposed MOERA algorithm.

Theorem 2. *MOERA produces feasible solutions to* $\mathbb{P}_0$ *with a competitive ratio* $r = 1 + \gamma n_0$.

Remark. The above result is reasonably good as it can be observed that r is monotonically decreasing with the parameters ε_1 and ε_2, and n_0 can be small, meaning that each user is limited to access a small subset of the edge clouds due to hardware or software specifications or data privacy concerns for example. We will further evaluate the empirical competitive ratio of the algorithm with real-world data in the next section.

6 Evaluation

We built a discrete-time simulator in Python to validate the performance of MOERA. We conducted experiments using both real-world and synthetic data and we report the experimental results in this section. All the measurements were performed on a Linux server equipped with Intel Xeon CPU E5-2687W (3.0 GHz) and 512 GB of RAM. We modeled the linear and convex programs by Pyomo and solved them by invoking IPOPT.

6.1 Datasets and Experimental Settings

We use various datasets obtained from different real-world application scenarios to validate the performance of MOERA. In the following, we describe these datasets in detail and provide the experimental settings that will be used throughout our evaluations.

Rome Taxi Dataset. The first real-world dataset we use is the Roma taxi trajectory traces [23]. This dataset contains the trajectory of taxis in the city of Rome over one month. To represent an edge computing application scenario, we envision that an edge cloud system would be deployed in the center area of Rome city with 15 edge clouds that are located at 15 selected metro stations, as shown in Fig. 4a. The edge clouds in the system will be used by the customers (termed as users hereafter) sitting in taxis, whose mobility patterns are provided by the trajectories of the taxis. The number of users varies from hour to hour but is generally around 300 in the dataset. We collect the GPS locations for the 15 edge clouds (i.e., metro stations) manually on Google Maps. We average the locations of the users within each minute to generate per-minute location data points for each user.

Darmstadt Kraken Dataset. The kraken.me project [24] aims to analyze user behavior and enable personal assistance for its users. Part of the project includes an Android application to collect and track user trajectory and other behaviors. Over the course of several weeks, we collected more than 26 million unique location data points from about 200 users. In this application scenario, we consider using the location of home routers for potential edge cloud locations. This is due to the fact that home routers are ubiquitously available in urban areas and the feasibility of using home routers as cloudlets or edge cloud discovery brokers has already been confirmed [2], [25].

We captured the location of WiFi access points in the city of Darmstadt, Germany with the help of a mobile application that senses WiFi signals as well as the information about the network. Participants walked around the city and collected a total of 23,744 access points. The exact location of the access point is estimated from multiple measurements of the same access point using trilateration. We eliminate duplicate access points based on the BSSID/MAC address of the devices. Then, by doing a vendor lookup for the addresses, we eliminate all manufacturers that do not produce home routers. While this might still lead to some wrong data (i.e., devices that are not home routers), we still argue that overall, the data gives us a good indication of the number of routers available in an urban area. From the remaining data, we select 50 routers to act as edge clouds. The locations of these edge clouds are publicly available [26]. We place those manually as shown in Fig. 4b. The placement is done such that we cover areas populated by the users of kraken.me, for instance we place multiple routers close to their homes and workplaces. Since most of the kraken.me users were students, we know the locations of the university and the student dorms. Other routers are placed in between those areas to model transitions that will occur whenever users are on the move.

We now generate the trajectory input data for the evaluation as follows: We choose the most active day (being it with the most mobile users) and we consider the time between 1 to 8 pm (7 hours). We divide the time period into one-minute time slots and for each of those time slots we obtain the edge cloud (i.e., the router) that is closest to the reported position of this user at that particular time. If there is no up-to-date location data in one particular time slot, we assume the position stays the same. We do however count the number of those "inaccurate" positions that occur for each user within a day. If it is greater than 50 percent of all time slots, we discard the user completely. In case there is more than one position update in the time slot, we average the positions.

User Workload. To understand the impact of the distribution of *user workload* on the effectiveness of MOERA, we use three different workload distributions: uniform, normal, and power-law. The power-law distribution represents highly skewed workload, which can be observed in typical online social network services, where the number of friends of each user on the social network satisfies the power law. For all the distributions, we first fix a base workload as `base`. For the uniform distribution, the workload is generated in the range of $[\omega, 2 \times \omega]$ uniformly at random. For the normal distribution, we set the average as ω and the variance as 0.5ω with the negative tail cut. For the power-law distribution, we draw samples in $[0, 1]$ from a power distribution with probability density function $P(x, \omega) = \omega x^{\omega-1}$ with positive exponent $\omega - 1$. Note that the absolute value of ω does not affect the performance of the algorithms as we will set the capacity of the edge clouds according to the total generated workload in the system.

The total capacity of the edge clouds is assumed to be slightly larger than the total workload in the system by design. More specifically, we assume that the overall utilization of the system keeps at the level of 80 percent. As a result, the total capacity is set to be 1.25 times the total workload. The *capacity* will be distributed to all the edge clouds proportionally to the frequency of users being attached to them, i.e., the total number of direct user connections in all the relevant time slots.

Edge Cloud Prices. We generate the *operation price* as follows: For each edge cloud, we first determine its base operation price reversely proportional to its capacity. This is reasonable due to the economy-of-scale effect on both energy and maintenance. The real-time operation price for each edge cloud follows Gaussian distributions, where we set the the mean value as the base price we just generated and the standard deviation as half of the base price [10]. The network delay is used to calculate the service quality cost and for each user the network delay can be partitioned into two parts: the delay between the user and the access point and the weighted average delay between the access point and the recruited edge clouds by the user. The delay in our model is measured by the geographical distance between any two entities based on their GPS locations. The *service quality price* is set to be proportional to the measured delay. The migration cost is associated with the bandwidth price and the bandwidth usage during the migration. In our model the *bandwidth price* is not assumed to be time-varying. However, different edge clouds may connect to the Internet via different Internet providers. We categorize all the edge clouds in three clusters, each of which is subscribed to one of the three Internet providers: Tiscali Italia, Vodafone Italia, and Infostrada-Wind. The per-month flat rate prices averaged for 1Mbps connection are 2.49 Euro, 4.86 Euro, and 1.25 Euro, respectively [27]. We will use the relative ratios between these prices to set the bandwidth prices for the three categories of edge clouds. We generate the *reconfiguration price* following a Gauss distribution with a cutoff of the negative tail.

6.2 Empirical Competitive Ratio

The theoretical analysis has already proven an upper bound on the competitive ratio for the online algorithm. We now validate how MOERA would perform in reality. We carry out experiments using the above settings and we compare the results of MOERA with two groups of algorithms: atomistic and holistic. Atomistic algorithms only consider the static part in the total cost and they include:

- The `perf-opt` algorithm aims at minimizing only the service quality cost E_Q in every time slot.
- The `oper-opt` algorithm minimizes only the operation cost E_O in each time slot.
- The `stat-opt` algorithm minimizes the total static cost $E_O + E_Q$ in each time slot and ignores the dynamic costs for reconfiguration and migration.

The algorithms in the holistic group include:

- The `offline-opt` algorithm minimizes P_0 assuming a global view on all the time slots in advance. This is considered impractical and only serves as a baseline.
- The `online-greedy` algorithm directly minimizes the objective value of $\mathbb{P}_0$ in every time slot. Decision making is based on the outcome of the previous time slot, but considers no future information.

Single-Objective versus Multi-Objective. The experimental results are shown in Fig. 5. From the Roma taxi traces, we select the data from date Feb 12, 2014 and we choose six hours from 3 PM through 9 PM as six independent test cases. We set the length of a time slot to one minute and thus each of the test cases consists of 60 time slots. All the values are normalized by the offline optimal objective. The experiments are repeated independently for five times and the plots show the mean values as well as the standard deviations. As we can see from the figure that the algorithms from the atomistic group perform poorly as expected. Among them, the `perf-opt` performs the best, thanks to the reduced frequency of workload migration because of the moderate mobility in the Roma taxi dataset. The `online-greedy` algorithm in the holistic group performs better than any of the atomistic algorithms. However, we still notice a considerable gap to the offline optimal, which is mainly due to the reasons we already discussed at the end of Section 2. In contrast, our online algorithm (denoted as `online-moera`) can produce near-

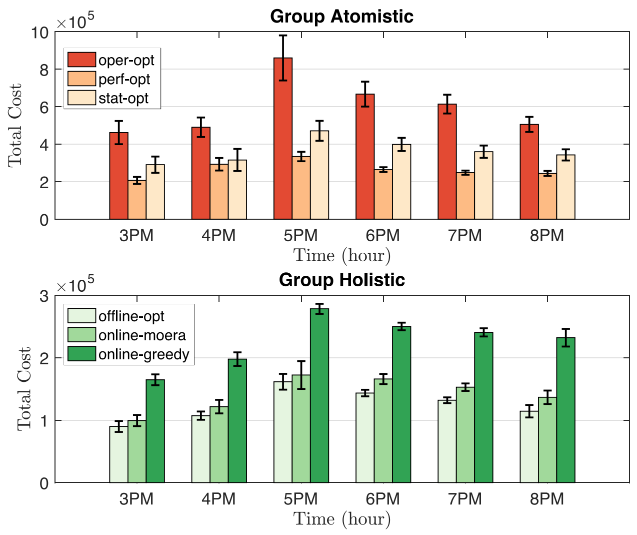


Fig. 5. Performance comparison among the two groups of algorithms with user workloads generated following a power-law distribution in the Rome Taxi scenario.

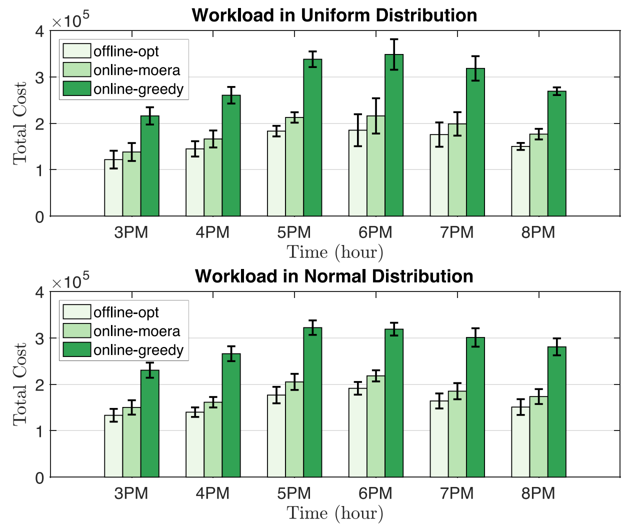


Fig. 6. The performance of MOERA compared with the optimal offline and the greedy solutions under uniformly and normally distributed user workloads in the Rome Taxi scenario.

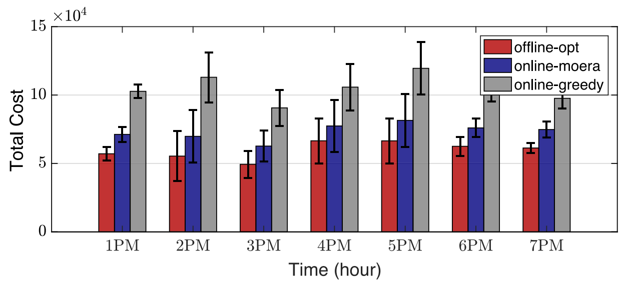


Fig. 7. Performance comparison with user workloads generated following a power-law distribution in the Darmstadt Kraken scenario.

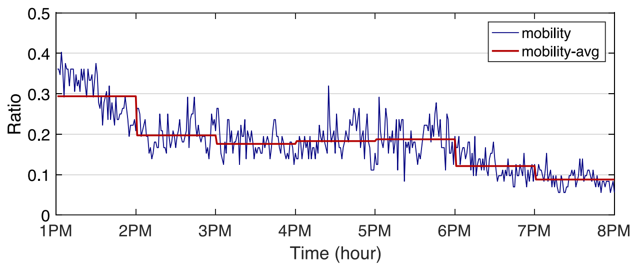


Fig. 8. Mobility level during the selected period of time in the Darmstadt Kraken scenario.

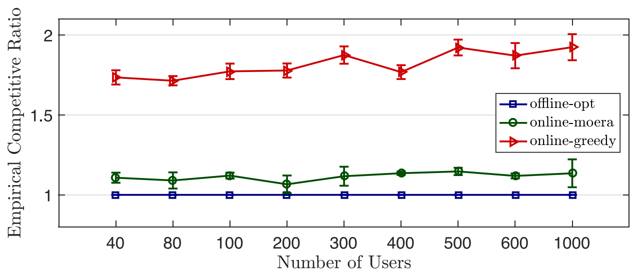


Fig. 9. Performance comaprison with user mobility generated following a random-walk process.

optimal results, achieving an improvement of up to 60 percent compared to the online greedy algorithm.

Different Workload Distributions. Fig. 6 illustrates the performance of our algorithm under different workload scenarios, where we generate the user workload using uniform and normal distributions in addition to the power-law distribution. As we can see that MOERA preserves similar properties, i.e., producing near-optimal solution and up to 70 percent improvement compared to `online-greedy`, under any of the workload distributions and MOERA performs even slightly better under uniform workloads.

Different Mobility Levels. Fig. 7 shows the performance of MOERA in the Darmstadt Kraken scenario. We plot the results that are obtained with the trajectory data collected from 1 PM through 8 PM (7 hours) on Feb 7, 2015. The total number of users is 72 in the selected dataset. As we can see that MOERA achieves an empirical competitive ratio smaller than 1.2, while it is around 1.6 for `online-greedy`. The reason that the improvement of MOERA is less significant than that in the Rome Taxi scenario is that we have significantly less users. To examine the correlation between the empirical competitive ratio and the mobility pattern, we plot the mobility level measured by the ratio between the number of users that have moved and the total number of users in Fig. 8. It is easy to observe that the performance of `online-greedy` shows a strong correlation with the mobility pattern, while the performance of MOERA remains relatively stable regardless of the level of mobility.

Synthetic Mobility Patterns. Fig. 9 illustrates the experimental results with synthetic mobility data under various numbers of users in the Rome Taxi scenario, which is used to validate the generality of our algorithm. The synthesis mobility data is generated following a random walk process: We assume each user starts from an arbitrary metro station equipped with an edge cloud and is traveling with the metro. In each time slot, each user determiners their location for the next time slot by choosing randomly from the neighbor stations with an edge cloud equipped or just staying at the same metro station. Assume in a certain time slot the user is at a location with three neighbors so the probably of moving to any of the three neighbors, as well as of staying at the same location, in the next time slot, would be 25 percent. Following the above process we generate the movement traces of the users. We vary the number of users from 40 to 1,000 and we compare our algorithm with the `offline-opt` and `online-greedy` algorithms.

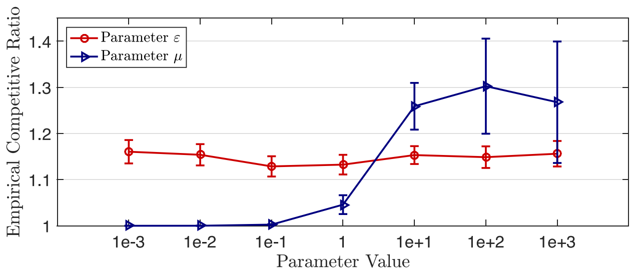


Fig. 10. The impact of the parameter ε and μ on the empirical competitive ratio.

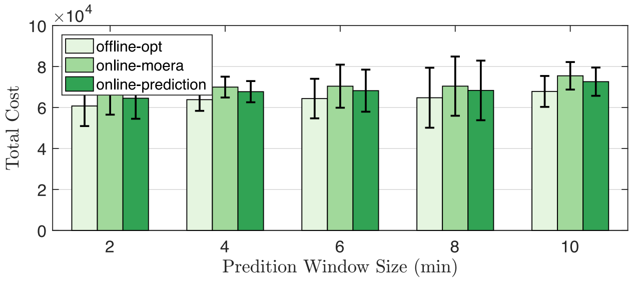


Fig. 12. Comparison between MOERA and a prediction-based approach.

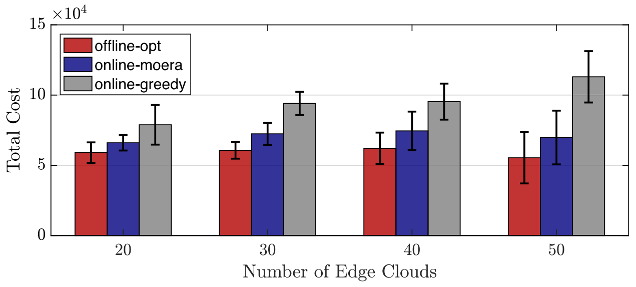


Fig. 11. Performance of MOERA when restricting the access of each user to different numbers of edge clouds.

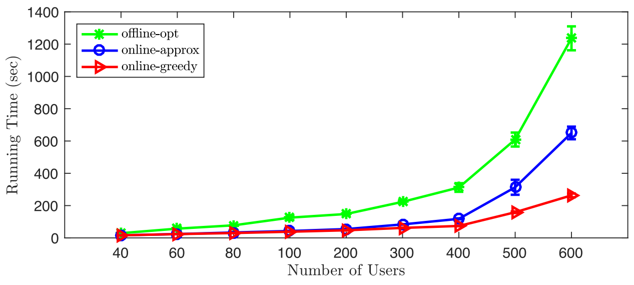


Fig. 13. Running time comparison.

We observe that our algorithm performs in a similar way as in the real-world mobility scenario, i.e., the empirical competitive ratio is around 1.1, which is very close to the optimal, while the `online-greedy` has empirical competitive ratios up to 1.8. In addition, our algorithm performs stably regardless of the number of users.

6.3 Impact of Parameters

Algorithm Parameters. Fig. 10 shows the impact of the parameters ε_1 and ε_2 on the performance of our algorithm in the Rome Taxi scenario. We set $\varepsilon_1 = \varepsilon_2 = \varepsilon > 0$ and we vary ε from 10^{-3} to 10^3 in a logarithmic scale in all the above test cases. It is interesting to notice that with the increase of ε, the empirical competitive ratio of our algorithm declines slightly at the beginning and then increases to a stable level. We report also in Fig. 10 the impact of the ratio between the weight of the dynamic cost and the weight of the static cost (denoted as μ) in the objective by varying its value from 10^{-3} to 10^3 in a logarithmic scale. We observe that when μ is small, i.e., the dynamic cost is negligible, our algorithm can roughly achieve optimal results. When the dynamic cost dominates, our algorithm can still achieve a stable yet reasonably good competitive ratio.

Proportion of Edge Clouds. Fig. 11 depicts the performance comparison when we restrict the access of each user to different numbers of edge clouds in the Darmstadt Kraken scneario. The x-axis represents the number of edge clouds we allow for each user to access, varying from 20 through 50 with a step of 10. The subsets of the edge clouds are selected uniformly at random. In general, we can observe that the performance of MOERA is relatively stable, with an empirical competitive ratio of around 1.18, confirming that MOERA scales well to large problem instances.

6.4 Comparison with Prediction-Based Approaches

One of the main advantages of MOERA is that the algorithm does not require any a priori information for decision making. To validate the effectiveness, we also compare MOERA against a prediction-based approach, described as follows: In every time slot, we assume that the prices as well as the user location for the time window with w future time slots are available and we compute the optimal resource allocation decision for the $w + 1$ time slots. Note that the prediction method is out of the scope of this paper and thus, we simply assume that perfect knowledge of the future can be obtained. The resource allocation decision for the current time slot is kept in the final solution, while the decision for other time slots are directly abandoned. We then repeat this process for all the considered time slots. We apply this approach on the Rome Taxi dataset and use the same parameters to generate the user workload using the power distribution. The results with varying window sizes are depicted in Fig. 12. As we can see that with the increase of the window size, the prediction-based approach achieves better results due to the fact that more knowledge of the future is available and thus, the decision is more accurate. MOERA performs slightly worse than the prediction-based approach as expected, but the difference is within 5 percent compared with the prediction-based approach. This confirms that MOERA can achieve a quite good performance even without requiring a priori knowledge of the system, making it more practical.

6.5 Running Time

We compare the running time of MOERA to the optimal offline solution and to the online greedy solution using the Rome Taxi dataset under different numbers of users. The experimental results are show in Fig. 13. Note that the values in the figure are the combined running time for 60 independent time slots in one hour. It can be seen that MOERA achieves a significant reduction (more than 2x) on running time and keeps at a comparable level as the greedy approach.

7 Discussion

We now discuss some practical issues in implementing the proposed algorithm in real systems. One important issue is on scalability. As a limitation, the proposed algorithm is centralized at this moment. This is due to the fact that future edge computing systems are expected to be managed in a logically centralized manner with a global system view so that the desired flexibility and efficiency that we already have in cloud computing can be preserved [28]. Nevertheless, we notice that several distributed approaches such as decomposition techniques [29] can be applied to solve the required convex programming in MOERA in a more distributed manner and thus, the scalability issue of MOERA can be mitigated. Another important issue is how to obtain the user workload information for decision making. As we do not assume any knowledge of the future in terms of user workload and other system parameters, the algorithm only requires the user workload information for each time slot at the beginning of the time slot. This workload information can be generated through a daemon module running with the mobile application that carries out analyses for the computational requests of the mobile application and be submitted to the edge computing platform through some well-defined interfaces. The last issue is on the workload distribution among the edge nodes. It is envisioned that modern mobile applications that are compatible with edge computing will be refactored based on the microservice architecture paradigm [30]. With this microservice-based architecture, application requests can be partitioned and served by multiple instances of microservices on different edge nodes. Each instance of a microservice can be scaled separately by assigning an appropriate amount of resources to it.

8 Related Work

The concept of edge computing was initially inspired by the idea of deploying computing servers at the network edge to enhance the performance of mobile devices [5], [31]. While numerous novel architectures for edge computing [6], [32], [33], [34], [35], [36] have been proposed, the resource allocation problem in such systems remains as a critical challenge.

Single-Cloudlet Task Offloading. Much of the existing research in this area is on allocating edge cloud resources to computational tasks offloaded from mobile devices. COSMOS [37] is a system that efficiently manages cloud resources for offloading requests to both improve the mobile performance and reduce the provider's monetary cost. Deng et al. [38] study online scheduling policies to maximize data offloading under unpredictable user mobility patterns. Chen et al. [39] focus on game-theoretical mechanisms for offloading decision making in the presence of multiple users, taking into account the energy consumption and the delay. Hou et al. [20] study the reconfiguration in edge clouds and propose an efficient online algorithm for configuration updating. However, all of them are focused on resource allocation in a single edge cloud environment.

Multi-Cloudlet Resource Management. On the other hand, attentions have been paid very recently on resource management in an edge cloud computing system with multiple edge clouds. Jia et al. study the optimal placement of cloudlets in wireless Metropolitan Area Networks and design an algorithm for user to cloudlet allocation [40]. In a follow-up work, they further propose an efficient algorithm for load balancing among multiple edge clouds [41]. Mukherjee et al. proposed an optimal cloudlet selection strategy to reduce power and latency in multi-cloudlet environments [42]. Wang et al. study the problem of joint task assignment and scheduling in mobile edge clouds by considering both the data movement and processing [43]. Recently, Jiao et al. explore the online control of both the cloudlets and the servers inside the cloudlets to operate the distributed cloudlet system towards the optimal cost [44]. Wang et al. study the service placement problem for supporting social virtual reality applications in edge computing [45]. The most relevant works to ours are from Wang et al. [17] and Urgaonkar et al. [13], where they propose stochastic frameworks for dynamic workload migration based on Markov Decision Processes (MDPs) and the Lyapunov optimization technique. However, all of the work does not include the reconfiguration of edge clouds in the cost model and either requires statistic information on the user mobility pattern or assumes a Markov chain model for user movement, which is not necessary in our model.

Resource Management in Geo-Distributed Clouds. There is also research on workload distribution and resource allocation in geo-distributed data centers [10], [11], [15]. While sharing some common objectives with our problem, they are intrinsically different from edge computing environments as neither delay sensitivity nor user mobility is considered in their models.

Summary. In contrast to existing work, our study addresses the challenge of allocation and continuous adaptation of resources in edge clouds, accommodating arbitrary resource price and user mobility dynamics. Our model captures multiple types of important costs, including static and dynamic ones; our online algorithm, without any knowledge on the future, makes resource allocation decisions on the fly while guaranteeing a parameterized competitive ratio for the worst-case inputs.

9 Conclusion

In this paper, we studied the online resource allocation problem in edge cloud systems. We identified the major challenges and further captured all of them by a comprehensive model, where we incorporated as the optimization objective the costs associated with edge cloud operation, delay, server reconfiguration, as well as service migration. We proposed MOERA, a mobility-agnostic online algorithm that can guarantee a parameterized competitive ratio. The effectiveness of the algorithm was also validated by extensive experiments using both real-world and synthetic data. A research gap left in our paper is to incorporate dynamics on user arrival and departure. However, due to the nondeterministic number of variables to be decided for resource allocation, the problem becomes very hard and it is still not clear if it is possible to obtain an online algorithm with strict theoretical guarantee. We leave this for future exploration.

Acknowledgments

The authors thank the anonymous reviewers for their constructive comments. This work was partially funded by the German Research Foundation (DFG) under Grant No. 392046569, the National Natural Science Foundation of China (NSFC) under Grant No. 61761136014, the DFG Collaborative Research Center (CRC) 1053 – MAKI, and partially by the US National Science Foundation under Grant No. 1564348. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] N. Fernando, S. W. Loke, and J. W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[2] J. Gedeon, C. Meurisch, D. Bhat, M. Stein, L. Wang, and M. Mühlhäuser, "Router-based brokering for surrogate discovery in edge computing," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, 2017, pp. 145–150.

[3] E. Cuervo, K. Chintalapudi, and M. Kotaru, "Creating the perfect illusion: What will it take to create life-like virtual reality headsets?" in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl.*, 2018, pp. 7–12.

[4] Accenture Consulting, "Multi-access edge computing for pervasive networks," 2018. [Online]. Available: https://accntu.re/2tRAF0f, Accessed: Jun. 28, 2018.

[5] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[6] J. Cho, K. Sundaresen, R. Mahindra, J. V. der Merwe, and S. Rangarajan, "ACACIA: Context-aware edge computing for continuous interactive applications over mobile networks," in *Proc. 12th Int. Conf. Emerging Netw. Exp. Technol.*, 2016, pp. 375–389.

[7] N. Panwar, S. Sharma, and A. K. Singh, "A survey on 5G: The next generation of mobile communication," *Phys. Commun.*, vol. 18, pp. 64–84, 2016.

[8] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1281–1290.

[9] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.

[10] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed online resource allocation in multi-tier distributed cloud networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2556–2570, Aug. 2017.

[11] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, A. Sala, and J. Li, "Online control of cloud and edge resources using inaccurate predictions," in *Proc. IEEE/ACM Int. Symp. Quality Service*, 2018, pp. 1–6.

[12] N. Buchbinder, N. Jain, and I. Menache, "Online job-migration for reducing the electricity bill in the cloud," in *Proc. Int. Conf. Res. Netw.*, 2011, pp. 172–185.

[13] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. S. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, 2015.

[14] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3330–3345, Dec. 2015.

[15] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 22–31.

[16] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 1350–1354.

[17] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. S. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf.*, 2015, pp. 1–9.

[18] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, 2015.

[19] S. Wang, R. Urgaonkar, K. Chan, T. He, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[20] I. Hou, T. Zhao, S. Wang, and K. Chan, "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2016, pp. 291–300.

[21] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 436–444.

[22] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3/4, pp. 231–357, 2015.

[23] Roma taxi dataset. (2014). [Online]. Available: http://crawdad.org/roma/taxi/20140717/, Accessed: Sep. 11, 2018.

[24] I. Schweizer and B. Schmidt, "Kraken.me: Multi-device user tracking suite," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.: Adjunct Publication*, 2014, pp. 853–862.

[25] C. Meurisch, A. Seeliger, B. Schmidt, I. Schweizer, F. Kaup, and M. Mühlhäuser, "Upgrading wireless home routers for enabling large-scale deployment of cloudlets," in *Proc. Int. Conf. Mobile Comput. Appl. Services*, 2015, pp. 12–29.

[26] Darmstadt AP locations. (2014). [Online]. Available: https://fileserver.tk.informatik.tu-darmstadt.de/SUN/darmstadt_ap.dat, Accessed: Sep. 11, 2018.

[27] Roma network prices. (2014). [Online]. Available: http://www.tempobox.it/en/index.htm, Accessed: Sep. 11, 2018.

[28] M. Satyanarayanan, "The emergence of edge computing," *IEEE Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[29] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[30] A. Reznik, R. Arora, M. Cannon, L. Cominardi, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella, C. Turyagyenda, and Z. Zheng, "Developing software for multi-access edge computing," 2017. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp20_MEC_SoftwareDevelopment_FINAL.pdf, Accessed: Mar. 5, 2018.

[31] 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 Comput.*, vol. 14, no. 2, pp. 24–31, Apr.–Jun. 2015.

[32] M. Chen, Y. Hao, Y. Li, C. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6-Supplement, pp. 18–24, Jun. 2015.

[33] A. Bhattcharya and P. De, "Computation offloading from mobile devices: Can edge devices perform better than the cloud?" in *Proc. 3rd Int. Workshop Adaptive Resource Manage. Scheduling Cloud Comput.*, 2016, pp. 1–6.

[34] R. Stoenescu, V. A. Olteanu, M. Popovici, M. Ahmed, J. Martins, R. Bifulco, F. Manco, F. Huici, G. Smaragdakis, M. Handley, and C. Raiciu, "In-Net: In-network processing for the masses," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, pp. 1–15.

[35] M. Jang, H. Lee, K. Schwan, and K. Bhardwaj, "SOUL: An edge-cloud system for mobile applications in a sensor-rich world," in *Proc. IEEE/ACM Symp. Edge Comput.*, 2016, pp. 1–9.

[36] I. Burago, M. Levorato, and A. Chowdhery, "Bandwidth-aware data filtering in edge-assisted wireless sensor systems," in *Proc. 14th Annu. IEEE Int. Conf. Sens. Commun. Netw.*, 2017, pp. 1–9.

[37] C. Shi, K. Habak, P. Pandurangan, M. H. Ammar, M. Naik, and E. W. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 287–296.

[38] H. Deng and I. Hou, "Online scheduling for delayed mobile offloading," in *Proc. IEEE INFOCOM*, 2015, pp. 1867–1875.

[39] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[40] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct.–Dec. 2017.

[41] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

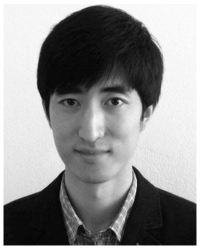
[42] A. Mukherjee, D. De, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *IEEE Trans. Cloud Comput.*, vol. 7, no. 1, pp. 141–154, Jan.-Mar. 2019.

[43] L. Wang, L. Jiao, D. Kliazovich, and P. Bouvry, "Reconciling task assignment and scheduling in mobile edge clouds," in *Proc. IEEE 24th Int. Conf. Netw. Protocols*, 2016, pp. 1–6.

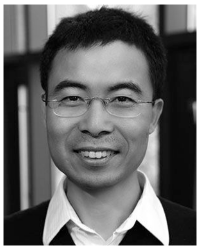
[44] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *Proc. 15th Annu. IEEE Int. Conf. Sens. Commun. Netw.*, 2018, pp. 1–9.

[45] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE INFOCOM*, 2018, pp. 1–9.

Lin Wang received the PhD degree in computer science with distinction from the Institute of Computing Technology, Chinese Academy of Sciences, in 2015. He was a visiting researcher with the IMDEA Networks Institute, Madrid, Spain, from 2012 to 2014, and a research associate with SnT Luxembourg from 2015 to 2016. Starting in July 2016, he has been head of the Smart Urban Networks Research Group, Telecooperation Lab, and has been an Athene Young investigator since 2018, at TU Darmstadt, Germany. His work has been published in IEEE INFOCOM, ICDCS, *JSAC*, *TPDS*, etc., and he has served as organizer or TPC member for various conferences such as IEEE CloudCom, ICCCN, ICC, IEEE/ACM DS-RT, and IEEE/IFIP NOMS. His current research interests include edge computing, networked systems, and energy-efficient algorithms.



Lei Jiao received the PhD degree in computer science from the University of Göttingen, Germany, in 2014. He was a researcher with IBM Research in Beijing, China, in 2010 prior to his PhD study. He was also a member of technical staff with Bell Labs in Dublin, Ireland, from 2014 to 2016. He is currently an assistant professor with the Department of Computer and Information Science, University of Oregon. His research interests include the models, algorithms, and analysis for the optimization and control of distributed systems and networks. His research has been published in journals such as the *IEEE/ACM Transactions on Networking* and the *IEEE Journal on Selected Areas in Communications*, and in conferences such as IEEE INFOCOM, ICNP, SECON, and IPDPS. He is on the technical program committee of a number of conferences including IEEE INFOCOM, IEEE/ACM IWQoS, IFIP Networking, and IEEE ICC.



Jun Li received the BS degree from Peking University, Beijing, China, in 1992, the ME degree from the Chinese Academy of Sciences, Beijing, China, in 1995, and the PhD degree (with honors) from the University of California, Los Angeles, California, in 2002, all in computer science. He is an associate professor with the Department of Computer and Information Science, University of Oregon, Eugene, Oregon, and directs the Network & Security Research Laboratory. Specializing in computer networks, distributed systems, and their security, he is currently researching Internet monitoring and forensics, Internet architecture, social networking, cloud computing, and various network security topics. He is a senior member of the Association for Computing Machinery (ACM). He is currently an editor of *Computer Networks* and chair of several workshops and symposiums. He has also served on several US National Science Foundation research panels and on more than 60 international technical program committees. He received a Presidential Scholarship for the ME degree and is a 2007 recipient of the NSF CAREER Award.

Julien Gedeon received the BSc and MSc degrees from Technische Universität Darmstadt, in 2013 and 2015, respectively. He is currently working toward the PhD degree in the Telecooperation Lab, Technische Universität Darmstadt, advised by Prof. Max Mühlhäuser. His research interests include in-network processing, edge computing, and smart cities.

Max Mühlhäuser received the PhD degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 1986. He is currently a professor and the dean of the Computer Science Department and the head of the Telecooperation Lab at TU Darmstadt, Germany. He was appointed adjunct professor with QUT Brisbane, in 2012 and has been a member of acatech, German National Academy of Science and Engineering, since 2015. Previously, he was the founder and head of DEC Research Labs Karlsruhe and he worked as either professor or visiting professor at universities in Germany, the US, Canada, Australia, France, and Austria. Now, he is the spokesperson of the Research Training Group (RTG) on Privacy and Trust for Mobile Users and acts as a deputy spokesperson of the Collaborative Research Center (CRC) on Multi-Mechanisms Adaptation for the Future Internet (MAKI). Further, he serves as a PI in the Center for Research in Security and Privacy (CRISP) and in the CRC on Cryptography-Based Security Solutions (CROSSING). He has published more than 400 articles, books, and book chapters. His research interests span widely across computer networks and distributed systems, human computer interaction, and cybersecurity, reliability, and trust.