Service Placement for Collaborative Edge Applications

Lin Wang, Lei Jiao, *Member, IEEE*, Ting He, *Senior Member, IEEE*,
Jun Li, *Senior Member, IEEE, ACM*, and Henri Bal

***Abstract*— Edge computing is emerging as a promising computing paradigm for supporting next-generation applications that rely on low-latency network connections in the Internet-of-Things (IoT) era. Many edge applications, such as multi-player augmented reality (AR) gaming and federated machine learning, require that distributed clients work collaboratively for a common goal through message exchanges. Given an edge network, it is an open problem how to deploy such collaborative edge applications to achieve the best overall system performance. This paper presents a formal study of this problem. We first provide a mix of cost models to capture the system. Based on a thorough formulation, we propose an iterative algorithm dubbed ITEM, where in each iteration, we construct a graph to encode all the costs and convert the cost optimization problem into a graph cut problem. By obtaining the minimum $s - t$ cut via existing max-flow algorithms, we address the original problem via solving a series of graph cuts. We rigorously prove that ITEM has a parameterized constant approximation ratio. Inspired by the optimal stopping theory, we further design an online algorithm called OPTS, based on optimally alternating between partial and full placement updates. Our evaluations with real-world data traces demonstrate that ITEM performs close to the optimum (within 5%) and converges fast. OPTS achieves a bounded performance as expected while reducing full updates by more than 67% of the time.**

***Index Terms*— Edge computing, service placement, performance optimization, approximation.**

I. Introduction

Edge computing has become a promising data processing solution for the Internet-of-Things (IoT) ecosystem thanks to its low-latency promises [2], [3]. Unlike large-scale cloud data centers, edge clouds are huge in quantity and are typically dispersed at the edge of the network [4]. Such proximity provides IoT applications a set of new opportunities including low-latency, reliable communication, early-stage data filtering, and privacy protection. Many edge applications, while reaping these benefits, require that multiple distributed clients work collaboratively for a joint task, sharing a common state that is replicated for and synchronized across all the clients [5], [6]. A typical example is multi-player gaming in augmented reality (AR) like Facebook Horizon [7] where a set of players (i.e., clients) at different geographical locations explore a common virtual world. Another example is federated machine learning where distributed clients collect observations and jointly train a machine learning model (with shared model weights) instead of gathering all the data to a central place for processing [8]–[10].

Edge computing has been deployed in various industries for handling the situation posed by resource limitation of local processing and large, unpredictable latency of remote cloud-based processing [11], [12]. According to the Open Edge Computing initiative [13], an edge cloud is able to offer compute/storage resources to any user in close proximity through an open, standardized interface, allowing a set of edge clouds in the same geographical region (e.g., a city) to form a shared edge resource pool. With the help of light-weight virtualization techniques [14], edge resources can be allocated at a fine granularity subject to quality-of-service (QoS) requirements and system-wide optimization goals.

In this paper, we focus on **collaborative edge applications** and we study the problem of service placement for such applications in the edge environment. As illustrated in Figure 1, a collaborative edge application consists of multiple dispersed clients and each client is composed of two parts: a service entity (SE) running on an edge node and a user entity (UE) running on the user device. The SE is defined as the bundle of the client data and the processing logic on the data. The SE takes care of the client state and computation-intensive tasks such as object recognition and tracking in the AR gaming case as well as the synchronization among clients. On the other hand, the UE is in charge of collecting client data and provides feedback to the client. The SE and its corresponding UE

Manuscript received March 6, 2020; revised July 23, 2020; accepted August 31, 2020; approved by IEEE/ACM Transactions on Networking Editor C. Joo. Date of publication October 5, 2020; date of current version February 17, 2021. This work was supported in part by the German Research Foundation (DFG) and the National Nature Science Foundation of China (NSFC) Joint Project under Grant 392046569 (DFG) and Grant 61761136014 (NSFC), and in part by the DFG Collaborative Research Center (CRC)–MAKI under Grant 1053. The work of Lei Jiao was supported by the Ripple Faculty Fellowship. The work of Ting He was supported in part by the U.S. Army Research Laboratory and in part by the U.K., Ministry of Defense under Grant W911NF-16-3-0001. The work of Jun Li was supported by the National Science Foundation under Grant 1564348. A preliminary version of this article appeared at IEEE INFOCOM 2018. *(Corresponding author: Lin Wang.)*

Lin Wang is with the Department of Computer Science, VU Amsterdam, 1081 HV Amsterdam, The Netherlands, and also with the Department of Computer Science, TU Darmstadt, 64289 Darmstadt, Germany (e-mail: lin.wang@vu.nl).

Lei Jiao and Jun Li are with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403 USA (e-mail: jiao@cs.uoregon.edu; lijun@cs.uoregon.edu).

Ting He is with the School of Electrical Engineering and Computer Science, Pennsylvania State University, State College, PA 16801 USA (e-mail: tzh58@psu.edu).

Henri Bal is with the Department of Computer Science, VU Amsterdam, 1081 HV Amsterdam, The Netherlands (e-mail: h.e.bal@vu.nl).

Digital Object Identifier 10.1109/TNET.2020.3025985

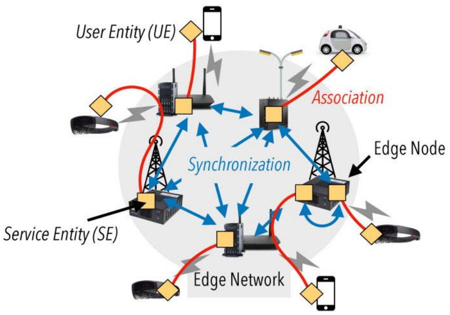


Fig. 1. Service placement problem for collaborative edge applications.

communicate to each other through some dedicated network protocols [15]. The service placement problem is to decide where to place the SE of each client in an edge network consisting of a set of edge nodes in order to achieve satisfactory QoS for the clients as well as economic (inexpensive) operations of the edge network. We call this problem the *collaborative edge service placement (CESP)* problem, which we deem to be fundamental for a rich set of collaborative edge applications.

The CESP problem is hard to solve and we have to consider the following conflicting factors: (1) Edge nodes are heterogeneous in terms of maintenance and operational costs and thus, achieving the best economic operations would push the SEs to edge nodes with lower operational costs; (2) SEs need to exchange data with their associated UEs and thus, placing an SE close to their UE could improve the QoS of the client with respect to communication latency; (3) SEs also need to exchange data with other SEs, e.g., synchronizing the state with other clients or completing a task with other clients collaboratively, and thus, placing SEs with frequent communication close to each other could result in better QoS for the clients of these SEs; (4) Since edge nodes are not intentionally designed to simultaneously accommodate a huge number of SEs, especially for edge applications that require specific hardware such as GPUs, resource contention needs to be well controlled, which suggests that each edge node should not be too crowded with SEs. The third factor is unique to CESP and all these intrinsically intertwined factors together complicate the problem.

Prior research efforts fall short of jointly considering all the above factors. On the one hand, most studies on edge computing are focused on computation offloading techniques and hardware/software architectures [16], [17]; edge resource management proposals generally lack the specific consideration of the state synchronization among clients as in the case of collaborative edge applications [18]–[26]. On the other hand, solutions for data placement across servers or distributed clouds for social networks address the optimization of network traffic or data storage, but neglect the impact of multiple important factors in the edge context such as activation cost and resource contention [27]–[31] which fundamentally change the settings and call for different approaches. While the general service placement problem has been extensively explored in various settings [32], [33], no existing algorithms are known to solve the CESP problem.

TABLE I

NOTATIONS

Symbol	Description
P	set of edge nodes
U	set of clients
U_p	set of clients with SEs placed on edge node p
p_u	edge node where the SE of client u is placed
p_u°	edge node attached to the AP that client u is connected to
$\mathbf{p}$	set of SE placement for all the clients, i.e., $\{p_1, ..., p_m\}$
$\mathbf{p}^q$	set of SE placement after expansion move on edge node q
f_u	comm. frequency between the UE and the SE of client u
$f_{u,v}$	comm. frequency from client u to client v
$d(p,q)$	network delay between edge nodes p and q
m_p	number of SEs on edge node p
$a(p)$	activation cost of edge node p
$b_u(p_u)$	cost of placing the SE for client u on edge node p_u
$c_1(p), c_2(p)$	parameters in co-location cost of edge node p
L	set of all interacting client pairs
$\mathbf{x}$	binary decisions in an expansion move, i.e., $\{x_1, ..., x_m\}$

In this paper, we present a formal study of the CESP problem. In particular, we make the following four contributions:

(1) We model the identified challenges with a comprehensive cost model for the edge network, based on which we formulate the CESP problem as a combinatorial optimization, which we prove is NP-hard.

(2) We transform the CESP problem and propose an efficient algorithm called ITerative Expansion Moves (ITEM) to solve it based on iteratively solving a series of minimum s-t cut problem instances.

(3) For the online version of CESP, we propose an online algorithm - OPTS, inspired by the optimal stopping theory. OPTS aims to optimize the alternation between ITEM and a tailored version of ITEM for incremental (partial) updates and minimizes the expected system cost under system stability cosntraints.

(4) We carry out extensive experiments with real-world data traces to validate the performance of the proposed algorithms. The results show that: ITEM performs very close to the theoretical optimal solution and outperforms the baselines by more than $2\times$, while exhibiting fast convergence. OPTS can reduce the frequency of full update (i.e., invocation of ITEM) by over 67%.

II. The Model

We model our system and introduce four types of cost, based on which we formulate the CESP problem. Table I lists the main notation we will use throughout the paper.

A. System Model

We are given an edge network consisting of a set of n edge nodes, denoted by $P = \{p_1, \ldots, p_n\}$, which are dispersed in a city. Each edge node is accompanied by an access point (AP) which allows the client to connect to the platform. We assume that hardware resources in the edge nodes are virtualized using container-based lightweight virtualization technologies and can thus be allocated and shared flexibly. All the edge nodes are connected to a wide area network (WAN) inside a city and the delay between each pair of edge nodes p and q is given by function $d(p,q)$ for $p, q \in P$.

We consider the problem of provisioning a collaborative edge application on the given edge network to serve m clients distributed in the city. An example scenario is illustrated in Figure 1. The set of clients is given by $U = \{u_1, \ldots, u_m\}$. Each client u has access to the edge network via its nearest AP, denoted by p_u°, in their vicinity.[1] For each client $u \in U$, an SE is brought up on one of the edge nodes p_u to handle the computation for client u (e.g., recognizing and tracking objects for client u in an AR application) within the collaborative edge application. The SE encapsulates all the necessary runtime environment as well as the service state for the client. The association traffic is measured by the frequency of communication between a client u and their SE p_u, which is denoted by f_u. In addition to employing the SE to process data, each client also synchronizes with other clients within the application, e.g., accessing the state of other clients or completing a task collaboratively with other clients. The synchronization traffic is measured by the frequency of interactions from client u to client v, which is given by $f_{u,v}$ ($f_{u,v} \triangleq 0$ if $u = v$) for $u, v \in U$. We denote by L the set of all interacting client pairs where $L = \{(u,v) \mid_{u,v \in U \wedge u \neq v}\}$.

B. Cost Models

We jointly consider multiple performance measures of the system. In particular, we model the following four cost types.

1) Activation Cost: For each of the edge nodes, if there is at least one SE being placed on it, a static activation cost has to be paid. Such an activation cost typically represents the cooling or other maintenance efforts in the edge node irrespective of the number of SEs being placed on the edge node [34]. The set of SEs that are placed on edge node p is denoted by $U_p = \{u \mid p \in P \wedge p_u = p\}$ and its cardinality is denoted by $m_p \in \mathbb{Z}^+$. Then, for each edge node $p \in P$ we define its monetary cost for maintaining its active state by $a(p)$ ($a(p) > 0$) if $m_p > 0$ and by 0 if $m_p = 0$, meaning that an edge node can be switched off (to save energy and maintenance costs) if it hosts no SEs. The edge cloud operator sets $a(p)$ according to the real maintenance cost for the edge node p. We introduce a variable $\delta_p \in \{0, 1\}$ where $\delta_p = 1$ if $m_p > 0$; $\delta_p = 0$ otherwise. The combined activation cost of all the edge nodes can be represented by

$$E_A = \sum_{p \in P} a(p)\delta_p. \tag{1}$$

2) Placement Cost: The placement cost is associated to the creation of SEs, which is incurred by the operation and communication of the servers for the SEs. For each client $u \in U$, the monetary cost for using the computing and storage resources when placing their SE on edge node p is characterized by $b_u(p_u) > 0$, which can vary among p and u due to the heterogeneity of the edge nodes and the difference in client demands. Denoting by $p_u \in P$ the placement decision for the SE of client u, the combined cost of placing all the SEs on the set of edge nodes can then be represented by

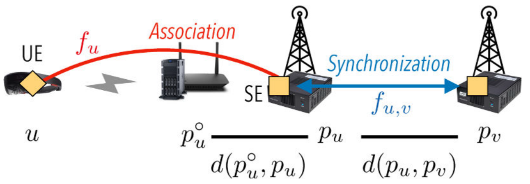


Fig. 2. Proximity cost includes the delay of the association traffic between the client and their SE and that of the synchronization traffic between clients.

$$E_B = \sum_{u \in U} b_u(p_u). \tag{2}$$

3) Proximity Cost: The proximity measure of a client contains two parts: how timely a client (or their UE) communicates with their SE and how timely a client communicates with other clients, as depicted in Figure 2. In order to give priority to frequent communications, we incorporate the communication frequency $f_u \geq 0$ between the UE and the SE of client u and the access frequency $f_{u,v} \geq 0$ from client u to client v. As a result, the combined proximity cost for the former is given by $\sum_{u \in U} f_u d(u, p_u)$, while for the latter it is given by $\sum_{(u,v) \in L} f_{u,v} d(p_u, p_v)$. We notice that the delay between a client and its AP is actually irrelevant to the placement decision. For the sake of simplicity we will omit it from our model and will use the following form for the total proximity cost in the rest of the paper:

$$E_D = \upsilon_D \sum_{u \in U} f_u d(p_u^\circ, p_u) + \upsilon_D \sum_{(u,v) \in L} f_{u,v} d(p_u, p_v), \tag{3}$$

where υ_D is a parameter that converts the proximity measure into monetary cost. This parameter should be defined according to the revenue loss of the cloud provider under the perceived latency [35]. Our model can also incorporate the transmission delay which is calculated as the exchanged data volume between any two clients divided by the network bandwidth between the two edge nodes hosting the two clients respectively. However, transmission delay is not regarded as a prority since the metadata exchanged between the different clients is usually small in size, e.g., few bytes or kilobytes for one synchronization cycle [36]. Therefore, we omit the transmission delay in our formulation for simplicity.

4) Co-Location Cost: The co-location cost is incurred by the resource contention among the SEs that are placed on the same edge node. This is considered unavoidable as edge nodes are not intentionally designed for large-scale resource multiplexing and perfect performance isolation is typically difficult with light-weight virtualization. The performance degradation due to SE co-location can be characterized by the "dilation" factor which was proposed to capture the performance of applications running on shared resources [37]. The dilation factor is defined as the ratio of the running time of the applications co-located on an edge node over the running time of an application running individually on the same edge node. According to Property 1 in [37], the dilation factor in our case can be approximated by the number of co-located SEs since the SEs are identical in the same application. Based on this,

[1] With a slight abuse of notation, we also use p_u°, i.e., the edge node that is directly attached to the access point, to denote the access point.

we use a general function $c_1(p)m_p + c_2(p)$ to characterize the monetary cost for the performance degradation of the co-located SEs given that performance degradation can also be induced by contention of resources other than CPU (e.g., memory, cache, or network) [38] which is represented by a constant $c_2(p)$. The model is general in the sense that the parameters $c_1(p)$ and $c_2(p)$ can be tuned to suit any practical needs depending on the edge node specification and the application type. Consequently, the total co-location cost in the system is characterized by

$$E_C = \sum_{p \in P} (c_1(p)m_p + c_2(p))\delta_p. \tag{4}$$

This cost can also be written with respect to a function $g(p_u)$ defined as $g(p_u) = c_1(p)$ if $p_u = p$ and zero otherwise, i.e.,

$$E_C = \sum_{u \in U} g(p_u) + \sum\nolimits_{p \in P} c_2(p). \tag{5}$$

Collectively, the above costs provide a comprehensive model for the overall system performance.

C. Problem Formulation

The overall performance of the application with SEs being placed is measured by the total system cost, i.e.,

$$E = E_A + E_B + E_D + E_C. \tag{6}$$

The CESP problem can now be formally defined as follows.

Definition 1: Given a collaborative edge application with a set U of clients running on an edge network consisting of a set P of edge nodes, the CESP problem is to decide the placement of all SEs of the application, i.e., determining p_u for all $u \in U$, such that the total cost, defined by E, of the edge network is minimized.

While the model does not impose hard resource constraints directly, hard constraints are considered in our algorithm design later. In Section III-F we show that hard constraints due to hardware (e.g., GPU) or security requirements can be modeled by restricting the number of SEs that can be placed to an edge node or restricting the placement to predetermined candidate set of edge nodes. The formulation is general enough to capture other service placement problems with similar affinity or anti-affinity constraints. However, the scope of the paper is limited to collaborative edge applications for clarity.

D. Hardness

Non-surprisingly, the problem is hard to solve and we show

Theorem 1 ***Hardness****: CESP is NP-hard.*

Proof: We conduct the proof through a polynomial-time reduction from the uncapacitated facility location (UFL) problem, which is known to be NP-hard [39]. The UFL problem is described as follows: Given a set of m customers, a set of n potential facilities, a cost function $e_{i,j}$ denoting the cost of shipping a product from facility i to customer j, a fixed cost r_i of opening facility i, the problem is to choose the facilities to open and the facilities to use to supply the m customers, in order to satisfy some fixed demand at minimum cost. Let us denote by x_i the decision variable for the opening of facility i, i.e., $\eta_i = 1$ if facility i is open; $\eta_i = 0$ otherwise. We also use $x_{i,j} = 1$ to denote that customer j is supplied by facility i; $x_{i,j} = 0$ otherwise. The UFL problem can be formulated with the following integer linear program.

$$\begin{aligned}(\mathbb{P}_1) \min & \sum_{i=1}^{n} r_i \eta_i + \sum_{i=1}^{n}\sum_{j=1}^{m} e_{i,j} x_{i,j} \\ \text{s.t.} & \sum_{i=1}^{n} x_{i,j} = 1, \forall j = 1, \ldots, m, \\ & x_{i,j} \le \eta_i, \forall i = 1, \ldots, n, j = 1, \ldots, m, \\ & \eta_i, x_{i,j} \in \{0,1\}, \forall i = 1, \ldots, n, j = 1, \ldots, m. \end{aligned} \tag{7}$$

The first constraint ensures that every customer will be supplied by a facility, while the second constraint ensures that only open facilities will be used to supply customers.

We now transform an arbitrary UFL instance to an instance of CESP. We treat the customers, facilities, costs r_i and $e_{i,j}$ as the clients, edge nodes, activation cost $a(p)$ and placement cost $b_u(p)$. Consequently, we obtain a special case for CESP where the proximity and co-location costs are all set to zeros. Such a transformation is obviously in polynomial time. Therefore, if we obtain an optimal solution to the CESP instance, an optimal solution to the UFL instance yields, which reaches a contradiction as UFL is NP-hard. □

The above complexity result reveals that any exact solution finding the optimum is impractical. Thus, we aim to design an efficient approximate algorithm in the following section.

III. Algorithm Design

Essentially, CESP requires minimizing a nonlinear function in a search space with a large number of dimensions. A general approach to exploring the optimum of such an optimization problem is starting from an arbitrary SE placement and iteratively improving the solution by local adjustments. While general-purpose optimization techniques such as simulated annealing can be employed, they require exponential time in theory and are extremely slow in practice. Our approach is also based on local adjustments, but we observe that *a set of optimal local adjustments can be simultaneously obtained by solving a graph cut problem*, rather than be carried out one by one or pairwise. As the graph cut problem can be efficiently solved using existing max-flow algorithms, the searching complexity is thus significantly reduced.

A. Algorithm Design Overview

We propose an efficient algorithm for the CESP problem based on ITeractive Expansion Moves (ITEM). The pseudocode of the algorithm is listed in Algorithm 1. An *expansion move* is defined as an optimization process where an edge node $q \in P$ is selected for "expansion" and variables p_u are simultaneously given a binary choice to either stay as $p_u^q = p_u$ or switch to edge node q, i.e., $p_u^q = q$ (q is thus expanded). Let $\mathbf{p}^q = \{p_1^q, \ldots, p_m^q\}$ denote the placement after one feasible expansion on edge node q with respect to current placement $\mathbf{p}$. Each accepted expansion move will strictly reduce the total cost $E(\mathbf{p})$ and the algorithm keeps

Algorithm 1 ITEM

```
flag ← true;
while flag = true do                        ▷ Search until convergence
    flag ← false;
    for $q \in P$ do                          ▷ Iterate over edge nodes
        $\mathbf{p} \leftarrow \{p_u \mid_{u \in U}\}$;        ▷ Current placement
        $\mathbf{p}^q \leftarrow$ EXPANSION($\mathbf{p}, q$);     ▷ Expansion move
        if $E(\mathbf{p}^q) < E(\mathbf{p})$ then            ▷ Improvement found
            $\mathbf{p} \leftarrow \mathbf{p}^q$;
            flag ← true;
return $\mathbf{p} = \{p_u \mid_{u \in U}\}$;
```

Algorithm 2 EXPANSION

```
construct an auxiliary graph $G$ w.r.t. edge node $q$;
obtain the minimum $s$-$t$ cut on graph $G$ using the state-of-the-art max-flow algorithm;
extract expansion decision $\mathbf{x}$ from the graph cut;
for $u \in U$ do                            ▷ Expanding edge node $q$
    $p_u \leftarrow q$ if $x_u = 1$;
return $\mathbf{p} = \{p_u \mid_{u \in U}\}$;
```

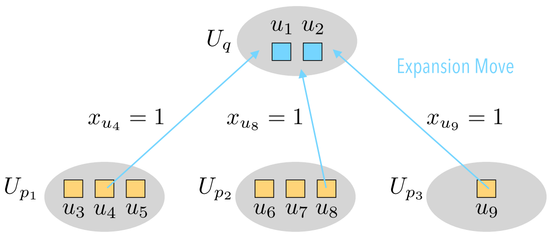


Fig. 3. Example expansion move where edge node q is selected for expansion.

searching by applying expansion moves iteratively over all the edge nodes until convergence. As the solution space is finite, ITEM must terminate after a finite number of iterations. Empirically, we have observed that in general it converges very fast (within five iterations), as shown in Figure 8.

The resultant placement $\mathbf{p}^q$ can be alternatively expressed by an indicator vector with binary decision variables $\mathbf{x} = \{x_1, \ldots, x_m\}$ where for all $u \in U$ we define $x_u = 1$ if $p_u^q = q$ and $x_u = 0$ otherwise. Note that if $p_u = q$ already, we always set $x_u = 1$. We denote by $E^q(\mathbf{x})$ the total cost corresponding to the new placement $\mathbf{p}^q$. The expansion move will compute an optimal $\mathbf{x}^*$ with the minimal $E^q(\mathbf{x}^*)$, from which the new placement $\mathbf{p}^q$ will be produced. A simple example is illustrated in Figure 3, where clients u_4, u_8 and u_9 switch to the selected edge node q from edge node p_1, p_2, and p_3, respectively, in the expansion move, while the other clients stay at their current edge node. Clients u_1 and u_2 will stay in edge node q according to the definition of expansion move. We have the following mappings. Note that the terms in circle represent that $x_u = 1$ if u is already on q in $\mathbf{p}$.

u	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
$\mathbf{p}$	q	q	p_1	p_1	p_1	p_2	p_2	p_2	p_3
$\mathbf{p}^q$	q	q	p_1	q	p_1	p_2	p_2	q	q
$\mathbf{x}$	①	①	0	1	0	0	0	1	1

It can be observed that the size of the solution space for $\mathbf{x}^*$ is 2^m and any brute-force search will result in exponential time complexity. We will show in the following sections that actually, $\mathbf{x}^*$ can be efficiently computed by encoding the total cost $E^q(\mathbf{x})$ in a graph and solving a corresponding graph cut problem leveraging existing max-flow algorithms. The state-of-the-art max-flow algorithms like the Orlin's [40] and KRT [41] can solve max-flow in time proportional to the product of the number of vertices and the number of links of the flow network in most cases.

B. Transforming the Objective Function

Given a current placement $\mathbf{p}$ and a selected edge node q, the objective of CESP after an expansion move can be expressed in terms of $\mathbf{x}$. We define the negation of x as $\bar{x}$, i.e., $\bar{x} = 1 - x$. The activation cost can be rewritten as follows.

$$\sum_{p \in P} a(p)\delta_p = \sum_{p \in P \setminus q} a(p)(1 - \prod_{u \in U_p} x_u) + \sigma_q, \tag{8}$$

meaning that the activation cost of p ($p \neq q$) is taken into account if and only if there exists at least one SE staying at p, i.e., $\exists u, x_u = 0$. Unfortunately, the product operation brings high-order terms in $\mathbf{x}$, which increases the complexity to the problem solving. However, we can actually transform this term into a sum of quadratic and linear terms by introducing an auxiliary variable z_p for each edge node p. For a particular edge node $p \in P \setminus q$, the transformation is

$$\prod_{u \in U_p} x_u = \max_{z_p \in \{0,1\}} (\sum_{u \in U_p} x_u z_p - (m_p - 1) z_p). \tag{9}$$

The final term σ_q in Eq. (8) is used to correct the case where edge node q does not host any SEs in the current placement $\mathbf{p}$. So this term incorporates the cost of q after the expansion move if q is being used in $\mathbf{p}^q$, i.e., $\sigma_q = a(q)(1 - \prod_{u \in U} \bar{x}_u)$.

Now, let us rewrite the the placement and the proximity costs. Observe that

$$\begin{aligned} b_u(p_u^q) &= b_u(q)x_u + b_u(p_u)\bar{x}_u, & (10)\\ d(p_u^\circ, p_u^q) &= d(p_u^\circ, q)x_u + d(p_u^\circ, p_u)\bar{x}_u, & (11)\\ d(p_u^q, p_v^q) &= d(p_u, q)\bar{x}_u x_v \\ &\quad + d(q, p_v)x_u\bar{x}_v + d(p_u, p_v)\bar{x}_u\bar{x}_v. & (12) \end{aligned}$$

Applying the above to (2) and (3) we can obtain

$$\begin{aligned} E_B^q(\mathbf{x}) &= \sum_{u \in U} (b_u(q)x_u + b_u(p_u)\bar{x}_u), & (13)\\ E_D^q(\mathbf{x}) &= \upsilon_D \sum_{u \in U} f_u(d(p_u^\circ, q)x_u + d(p_u^\circ, p_u)\bar{x}_u) \\ &\quad + \upsilon_D \sum_{(u,v) \in L} f_{u,v}(d(p_u, q)\bar{x}_u x_v \\ &\quad + d(q, p_v)x_u\bar{x}_v + d(p_u, p_v)\bar{x}_u\bar{x}_v). & (14) \end{aligned}$$

We now transform the co-location cost E_C^q. We notice that the second half of the co-location cost $\sum_{p \in P} c_2(p)\delta_p$ has the same form as the activation cost and we can apply exactly the same technique as explained above. Thus, we omit this part of cost in the following analysis for brevity. Denoting by m_p^q

the number of SEs being placed onto edge node p after the expansion move, the co-location cost is simplified as follows.

$$\begin{aligned} E_C^q &= \sum_{p \in P} c_1(p) m_p^q + \sum_{p \in P} c_2(p)\delta(p) \qquad (15) \\ &= \sum_{u \in U} c_1(q) x_u + \sum_{u \in U} c_1(p_u)\bar{x}_u + \sum_{p \in P} c_2(p)\delta_p. \end{aligned}$$

It is obvious that the first half of the co-location cost is linear in the x_u. The total cost in the objective after the expansion move can be expressed in terms of $\mathbf{x}$, i.e., $E^q = E_A^q + E_L^q + E_D^q + E_C^q$.

C. Optimizing Expansion Move via Minimizing Graph Cuts

We show that an optimal expansion move can be obtained by simply solving a graph cut problem. More specifically, we construct a helper graph and encode all the costs into weights on the graph edges. We then demonstrate that the min-cut of the graph corresponds to the optimal decisions for the expansion move.

1) Graph Construction: We now construct a graph G to encode the total cost E in our model. We first introduce m vertices, each of which corresponds to each client u. We then introduce n vertices to represent the set of edge nodes. Finally, we add a source vertex s and a vertex node t, where s corresponds to decision $x_u = 0$ and t corresponds to decision $x_u = 1$. As a result, the set of vertices in G is given by $\{u \mid_{u \in U}\} \cup \{p \mid_{p \in P}\} \cup \{s, t\}$. We start with encoding the activation cost. We first define an m-dimension indicator vector $\mathbf{y} = \{y_1, \ldots, y_m\}$ where $y_u = 1$ if $u \in U \setminus U_q$ and $y_u = 0$ otherwise. Then, we re-parameterize the right hand of Eq. (8) such that each quadratic monomial has exactly one complemented variable (e.g., $\bar{x}z$) with nonnegative coefficient, i.e.,

$$\begin{aligned} &\sum_{p \in P} a(p) + \sum_{p \in P \setminus q} (-a(p) z_p + \sum_{u \in U_p} a(p)\bar{x}_u z_p) \\ &\quad = \sum_{u \in U} y_u (a(p_u)\bar{x}_u z_{p_u} - a(p_u) z_{p_u}) + \sum_{p \in P} a(p). \qquad (16) \end{aligned}$$

For each $u \in U_p$, we add a link from u to p_u with weight $a(p)$. We also introduce a link from each p to the sink t with weight $a(p)$. This graph structure ensures that only weight of $a(p)$ will be included in the graph cut. The encoding of the correction term σ_q in (8) is analogous to the above but we use a simpler test-and-reject technique to handle this term [42]. We ignore this term during the expansion move and explicitly add it to E_A^q if there exists $u \in U$ such that $x_u = 1$. If the cost would increase, we reject the expansion move. Note that the second component in Eq. (15) is similar to the activation cost and can be handled exactly the same way. The weights on the links mentioned above become $a(p) + c_2(p)$.

For the other costs, we combine them all together and simplify them to the following form after some simple algebra.

$$\sum_{(u,v) \in L} \pi_{u,v} \bar{x}_u x_v + \sum_{u \in U^+} \tau_u x_u + \sum_{u \in U^-} |\tau_u| \bar{x}_u + \kappa. \qquad (17)$$

Note that we split $u \in U$ into two subsets U^+ and U^- where $\tau_u \geq 0$ if $u \in U^+$ and $\tau_u < 0$ otherwise. The symbol $\pi_{u,v}$ is the coefficient of *pairwise terms*, where

$$\pi_{u,v} = \upsilon_D f_{u,v}(d(p_u, q) + d(q, p_v) - d(p_u, p_v)), \qquad (18)$$

and τ_u is the coefficient for *unary terms*, where

$$\begin{aligned} \tau_u = {} & b(q, u) - b(p_u, u) + \upsilon_D f_u(d(p_u^\circ, q) - d(p_u^\circ, p_u)) \\ & + \upsilon_D \sum_{v \in U} (f_{u,v} d(q, p_v) + f_{v,u} d(q, p_u) - f_{u,v} d(p_u, p_v)) \\ & + c_1(q) - c_1(p_u), \qquad (19) \end{aligned}$$

and κ is a constant, which can be omitted from the graph construction as it has no impact on the expansion decisions. For each $u \in U^+$, we add a link from source vertex s to vertex u with weight τ_u. Similarly, for each $u \in U^-$, we add a link from vertex u to the sink vertex t with weight $|\tau_u|$. For each interacting client pair u, v, we add a link from u to v with weight $\pi_{u,v}$. The resultant graph G is illustrated in Figure 4.

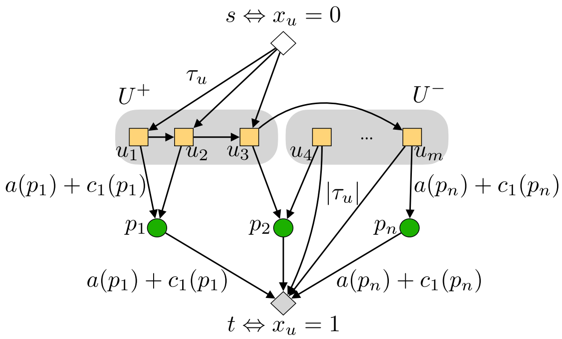


Fig. 4. Helper graph construction for encoding cost E^q.

The placement of SEs now can be obtained by computing the minimum s-t cut on G by employing the state-of-the-art max-flow algorithm [40], [41]. Specifically, we place the SEs after the expansion move according to the following policy.

$$p_u^q \triangleq \begin{cases} q & \text{if link } s\text{-}u \text{ is in the cut,} \\ p_u & \text{otherwise.} \end{cases} \qquad (20)$$

2) Correctness: We now analyze the correctness of the above approach. In particular, we show that

Theorem 2: Minimizing E^q is equivalent to obtaining the minimum s-t cut of graph G.

Proof: We first show the feasibility of obtaining the minimum s-t cut on the graph G. Through the graph construction we know that given an arbitrary E^q there always exists a graph G. However, the minimum s-t cut can be computed in polynomial time only if all the edge weights are nonnegative. For τ_u and $a(p) + c_2(p)$, this constraint follows in nature, while $\pi_{u,v} \geq 0$ is always satisfied which can be justified by the triangle inequality. Note that the network propagation delay is proportional to the physical distance and thus, $d(\cdot)$ is a metric and the triangle inequality holds since the physical distance is a metric in the Euclidean space.

Now we show the equivalence. For each edge node $p \neq q$, cost $a(p)$ is counted as long as there exists one link in set $\{(u, p) \mid_{u \in U, p \in P \setminus q}\} \cup \{(p, t)\}$ being contained in the cut, meaning that there exists $u \in U_p$ such that $x_u = 0$. Thanks to the auxiliary variable z_p, only link p-t will be included in the cut if there are more than one client that satisfies $x_u = 0$, contributing only a cost of $a(p)$ to E^q. For any pair of vertices u and v in graph G, the pairwise cost $\pi_{u,v}$ contributes to E^q if and only if $x_u = 0$ and $x_v = 1$. This corresponds to the case that link (u, v) is contained in the minimum s-t cut, where u is assigned to the s-component and v is assigned to the t-component. For any $u \in U^+$, the unary cost is counted if and only if the cut contains link s-u, meaning $x_u = 1$. Similarly, for any $u \in U^-$, the unary cost is counted if and only if the cut contains link u-t, meaning that $x_u = 0$. □

D. Approximation Ratio

We now provide a formal analysis to the performance of ITEM. Upon convergence, the performance gap between ITEM and theoretical optimum can be bounded by a parameterized constant as stated in the following theorem.

Theorem 3: Denote by constant $\epsilon = \sum_{p\in P} a(p)$ the maximal total activation cost achieved when all edge nodes in P are activated. Assume p'_u and p^*_u are the placement given by ITEM and the optimal placement of the SE for client u, respectively. We have $E(p'_u) \leq 2\lambda E(p^*_u) + \epsilon$, where

$$\lambda = \frac{\max_{p,q} d(p, q)}{\min_{p,q} d(p, q)}. \tag{21}$$

Proof: We first rearrange the total cost E into the following form with four different types of costs.

$$\begin{aligned} E &= E_A + E_B + E_D + E_C \\ &= \overbrace{\sum_{u\in U} b_u(p_u) + \sum_{u\in U} \upsilon_D f_u d(p^\circ_u, p_u) + \sum_{u\in U} g(p_u)}^{E_1} \qquad (22) \\ &\quad + \overbrace{\sum_{(u,v)\in L} \upsilon_D f_{u,v} d(p_u, p_v)}^{E_2} + \overbrace{\sum_{p\in P} (a(p) + c_2(p))\delta_p}^{E_3}, \end{aligned}$$

where E_1 is the unary cost which only depends on the SE placement of each single client, E_2 is the pairwise cost which depends on the placement of each pair of clients, and E_3 is the one-time occupation cost (consisting of the activation cost and part of the co-location cost) over all edge nodes. The final term is a constant which is irrelevant to the client SE placement and thus, we omit it from now on in the analysis.

We now select an arbitrary edge node $q \in P$ and define the set of clients that belong to q in the optimal solution p^*_u as $U_q \triangleq \{u \in U : p^*_u = q\}$. From the solution p'_u generated by ITEM, we can produce a new solution p^q_u by following one expansion move:

$$p^q_u = \begin{cases} q & \text{if } u \in U_q, \\ p'_u & \text{otherwise.} \end{cases} \tag{23}$$

As p'_u is a local optimum generated by ITEM with respect to expansion moves, such an expansion move will increase the total cost and thus, we have

$$E(p'_u) \leq E(p^q_u). \tag{24}$$

We now partition all edge nodes into three sets, i.e., interior, exterior, and boundary sets, with respect to U_q as follows:

$$U^I_q = U_q, U^O_q = U \setminus U_q, U^B_q = \emptyset.$$

Similarly, we partition all the client interactions into three sets with respect to U_q as follows:

$$\begin{aligned} L^I_q &= \{(u, v) \in L : u \in U_q, v \in U_q\}, & (25) \\ L^O_q &= \{(u, v) \in L : u \neq U_q, v \neq U_q\}, & (26) \\ L^B_q &= \{(u, v) \in L : u \in U_q, v \neq U_q\}. & (27) \end{aligned}$$

For simplicity, let us define $I_q \triangleq U^I_q \cup L^I_q$, $O_q \triangleq U^O_q \cup L^O_q$, and $B_q \triangleq U^B_q \cup L^B_q$. Let $E_4(\cdot)|_S$ denote a restriction of the summands of cost E to only the following terms

$$E_4(\cdot)|_S = E_1(p_u)|_{u\in U^S_q} + E_2(p_u, p_v)|_{(u,v)\in L^S_q}. \tag{28}$$

We can derive the following relationships.

$$\begin{aligned} E_4(p^q_u)|_I &= E_4(p^*_u)|_{I_q}, & (29) \\ E_4(p^q_u)|_O &= E_4(p'_u)|_{O_q}, & (30) \\ E_4(p^q_u)|_B &\leq \lambda E_4(p^*_u)|_{B_q}. & (31) \end{aligned}$$

The first equation follows because the SE placement for clients in U^I_q is the same for both solutions p^q_u and p^*_u. The second equation follows because the SE placement for clients in U^O_q is the same for both solutions p^q_u and p'_u. The last inequality is because $d(p^q_u, p^q_v) \leq \lambda d(p^*_u, p^*_v)$. Based on Eq. (24), we have

$$\begin{aligned} &E_4(p'_u)|_{I_q} + E_4(p'_u)|_{O_q} + E_4(p'_u)|_{B_q} + E_3(p'_u) \\ &\quad \leq E_4(p^q_u)|_{I_q} + E_4(p^q_u)|_{O_q} + E_4(p^q_u)|_{B_q} + E_3(p^q_u). \end{aligned} \tag{32}$$

Combined with Eq. (29), (30), and (31), we have

$$\begin{aligned} &E_4(p'_u)|_{I_q} + E_4(p'_u)|_{O_q} + E_4(p'_u)|_{B_q} + E_3(p'_u) \\ &\quad \leq E_4(p^*_u)|_{I_q} + E_4(p'_u)|_{O_q} + \lambda E_4(p^*_u)|_{B_q} + E_3(p^q_u). \end{aligned} \tag{33}$$

Consequently, we have

$$\begin{aligned} &\sum_{q\in P} \left(E_4(p'_u)|_{I_q} + E_4(p'_u)|_{B_q}\right) + E_3(p'_u) \\ &\quad \leq \sum_{q\in P} \left(E_4(p^*_u)|_{I_q} + \lambda E_4(p^*_u)|_{B_q}\right) + E_3(p^q_u). \end{aligned} \tag{34}$$

Note that when summarized over all the edge nodes, $\sum_{q\in P} E_4(p'_u)|_{I_q} = E_1(p'_u) + \sum_{q\in P} E_2(p'_u, p'_v)|_{(u,v)\in L^I_q}$ and $\sum_{q\in P} E_4(p'_u)|_{B_q} = 2E_2(p'_u, p'_v)|_{(u,v)\in\cup L^B_q}$. The former is trivial and the latter is because every pair $(u, v) \in L^B_q$ is calculated twice when iterating $q \in P$, once for (u, v) and the other time for (v, u). Similarly for p^*_u we have $\sum_{q\in P} E_4(p^*_u)|_{I_q} = E_1(p^*_u) + E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^I_q}$ and $\sum_{q\in P} \lambda E_4(p^*_u)|_{B_q} = 2\lambda E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^B_q}$. Therefore, we have

$$\begin{aligned} &E_1(p'_u) + E_2(p'_u, p'_v)|_{(u,v)\in\cup L^I_q} \\ &\qquad + 2E_2(p'_u, p'_v)|_{(u,v)\in\cup L^B_q} + E_3(p'_u) \\ &\quad \leq E_1(p^*_u) + E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^I_q} \\ &\qquad + 2\lambda E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^B_q} + E_3(p^q_u). \end{aligned} \tag{35}$$

Since it holds that

$$\begin{aligned} E(p'_u) &= E_1(p'_u) + E_2(p'_u, p'_v)|_{(u,v)\in\cup L^I_q} \\ &\quad + E_2(p'_u, p'_v)|_{(u,v)\in\cup L^B_q} + E_3(p'_u), & (36) \\ E(p^*_u) &= E_1(p^*_u) + E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^I_q} \\ &\quad + E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^B_q} + E_3(p^*_u), & (37) \end{aligned}$$

we have

$$\begin{aligned} E(p'_u) &\leq E(p^*_u) + (2\lambda - 1)E_2(p^*_u, p^*_v)|_{(u,v)\in\cup L^B_q} \\ &\quad - E_2(p'_u, p'_v)|_{(u,v)\in\cup L^B_q} + E_3(p^q_u) - E_3(p^*_u) \\ &\leq 2\lambda E(p^*_u) + \epsilon. \end{aligned} \tag{38}$$

The proof is completed. □

Note that in theory, $\min_{p,q} d(p, q)$ can be arbitrarily small, leading to a large λ. However, considering that edge nodes are typically evenly distributed on purpose for better coverage in a given geographical region, λ should be a reasonably small constant. Therefore, the approximate ratio is also bounded by a small constant in real scenarios.

E. Time Complexity and Convergence of ITEM

Theorem 4: ITEM can be finished in $O(\Delta(m^2(m + n) + n^2))$ time, where Δ is the number of iterations.

Proof: ITEM works iteratively. In each iteration, ITEM first constructs a flow graph with number of nodes $O(m+n)$ and number of vertices $O(m^2 + n)$. The number of vertices is the sum of the number of edge nodes and the number of clients, as depicted in Figure 4. The number of links is calculated as follows: For each each client, there is an link to either the source s or the sink t. So there are in total $O(m)$ source- or sink-client edges. The number of links between the clients is $O(m^2)$ in the worst case assuming every client interacts with all others. The number of links between clients and edge nodes is $O(m)$ as each client has a connection to one of the edge nodes. The number of links between edge nodes and the sink is given by $O(n)$. Putting all these together, the total number of links is bounded by $O(m^2 + n)$. After constructing the flow graph, ITEM applies the state-of-the-art max-flow algorithms [40], [41] to solve the min-cut problem on the constructed flow graph, with a time complex linearly related to product of the number of vertices and the number of links. Therefore, one iteration of ITEM can finish in time $O((m + n)(m^2 + n)) = O(m^2(m + n) + n^2)$. Consequently, the time complex of ITEM is $O(\Delta(m^2(m+n) + n^2))$ where Δ is the number of iterations. □

Theorem 5: ITEM is guaranteed to converge.

Proof: Let us assume the objective value produced by ITEM in iteration t is denoted by $E(t)$. It follows that $E(t) > E(t + 1)$ since ITEM aims to reduce the objective if possible in every iteration; otherwise ITEM will terminate. Thus, for all $\zeta > 0$ there always exists t such that $|E(t)-E'| < \zeta$ where E' is a constant between E^* and $2\lambda E^*+\epsilon$ as given by Theorem 3. Formally, we have $\forall\zeta > 0, \exists T \in \mathbb{N} : |E(t)-E'| < \zeta, \forall t > T$. This completes the proof. □

F. Handling Hard Placement Constraints

The co-location cost in the model already prevents an edge node from hosting too many SEs. Nevertheless, the proposed algorithm can be extended to the case where SE placement has some hard constraints due to hardware limitation or privacy concerns, e.g., certain SEs can only be placed on a subset of the edge nodes. During the expansion move (on edge node q for instance), we set $x_u = 0$ for all the clients that cannot be placed on q and we trim the objective function accordingly. Thus, solving the induced graph cut problem will only provide decisions for the rest clients and the placement constraints will be satisfied in the final placement generated by ITEM.

Our proposed algorithm can also be extended to the case where edge nodes have hard constraints on the resource capacity. For example, we can restrict the number of SEs placed on each edge node. Such a restriction can be incorporated in every expansion move where we restrict the number of SEs to be moved to the target edge node. The minimum s-t cut problem then becomes the minimum k-size s-t cut problem where the size of one of the graph components after the cut is restricted within k. While the minimum k-size s-t cut problem is NP-hard, Zhang has proposed an efficient algorithm which achieves a constant approximation ratio [43]. This algorithm can be adapted to solve our problem. Note that such hard capacity constraints on edge nodes can be heterogeneous among edge nodes as the constraint for each edge node is treated separately in expansion moves.

IV. Online Algorithm

This section focuses on extending ITEM to support online cases where clients are very likely mobile and placement decisions have to be adapted continuously.

A. Model Extensions

To properly characterize the online SE placement problem, we first introduce some extensions to our model. We consider a system where time is fragmented into time slots $t = \{1, 2, \ldots\}$ of equal length. Considering the nature of collaborative edge applications, a minimum time slot could be in length of minutes. Within each time slot, the system is considered static, i.e., clients remain at their current location. Clients will only change their location across time slots.

With the time dimension introduced, many of the parameters in our model can be made time-varying. For the ease of expression, we focus on the client location in particular. Other time-varying parameters can be easily incorporated. We use $p_{u,t}$ to denote the edge node that hosts the SE of client u at time t, $p^*_{u,t}$ to denote the location of client u at time t, $m_{p,t}$ to denote the number of SEs placed on edge node p at time t, and $\delta_{p,t}$ to denote the status of edge node p at time t. We assume the client mobility is random and the underlying distribution for $p^*_{u,t}$ is unknown a priori. Consequently, the total cost of the system at time t can be written as

$$\begin{aligned} E(t) &= E_A(t) + E_B(t) + E_D(t) + E_C(t) \\ &= \sum_{p\in P} a(p)\delta_{p,t} + \sum_{u\in U} b_u(p_{u,t}) + \sum_{u\in U} \upsilon_D f_u d(p^*_{u,t}, p_{u,t}) \\ &\quad + \sum_{(u,v)\in L} \upsilon_D f_{u,v} d(p_{u,t}, p_{v,t}) \\ &\quad + \sum_{p\in P} (c_1(p)m_{p,t} + c_2(p)\delta_{p,t}). \end{aligned} \tag{39}$$

While the above equation can be optimized separately in every time slot using the ITEM algorithm we proposed, one fact cannot be omitted, which is that the placement decision for a

Algorithm 3 INCU – Incremental Updates

```
1:  $t \leftarrow 1$;
2:  while true do
3:      $\hat{U}_t = \{u \in U \mid p^*_{u,t} = p^*_{u,t-1}\}$;
4:      $\check{U}_t = \{u \in U \mid p^*_{u,t} \neq p^*_{u,t-1}\}$;
5:      flag ← true;
6:      while flag = true do
7:          for $q \in P$ do
8:              $\mathbf{p}_t \leftarrow \{p_{u,t} \mid u \in U\}$;
9:              $\hat{\mathbf{p}}^q_t \leftarrow \{p_{u,t} \mid u \in \hat{U}_t\}$;        ▷ Static clients
10:             $\check{\mathbf{p}}^q_t \leftarrow$ EXPANSION($\mathbf{p}_t, \hat{\mathbf{p}}^q_t, q$);     ▷ Expansion
11:             $\mathbf{p}^q_t = \hat{\mathbf{p}}^q_t \cup \check{\mathbf{p}}^q_t$;
12:             if $E(\mathbf{p}^q_t) < E(\mathbf{p}_t)$ then       ▷ Improvement found
13:                 $\mathbf{p}_t \leftarrow \mathbf{p}^q_t$;
14:             else
15:                 flag ← false;
16:     $t \leftarrow t + 1$;
```

given SE may not be consistent across time slots. This brings up the issue of migrating SEs. When an SE of a client has to be migrated from one edge node to another, the experience of the corresponding client may be interrupted, waiting for the system to restore data and to sync states. For the clients that have changed their location across time slots the impact of this interruption may be under expectation. However, such an interruption can result in a very unpleasant experience to clients that remain static (referred to as "static clients" hereafter) and are continuously using the service.

B. Incremental Updates

Following the above argument we propose an algorithm for the online SE placement problem. The rationale behind the algorithm design is as follows: We must limit the interruption to the static clients as much as possible. Ideally, we do not allow the placement decision for static clients to change across time slots. This leads to the idea of incremental updates, where we simply fix the placement for the static clients in the expansion move and only consider the re-placement of the clients that have changed their locations applying the ITEM algorithm. We denote by $\hat{U}_t$ and $\check{U}_t$ the set of static clients and the set of clients that have moved at time t, respectively.

The pseudocode of the algorithm (namely INCremental Updates, INCU) is listed in Algorithm 3. The main procedure is similar to that of ITEM. At the beginning of a time slot t, we obtain client locations $p^*_{u,t}$ and calculate the sets $\hat{U}_t$ and $\check{U}_t$. Then, we carry out the iterative expansion moves to improve the placement as done in ITEM. Note that the function EXPANSION$(\mathbf{p}_t, \hat{\mathbf{p}}^q_t, q)$ in line 10 takes one more argument and is slightly different from its original form shown in Algorithm 2. Before we construct the auxiliary graph G_t, we fix all the values for $\hat{\mathbf{p}}^q_t$ in the objective function $E(t)$ and remove $\hat{\mathbf{p}}^q_t$ from the free variables. Consequently, the expansion move will only be applied on the variables $p^q_{u,t}$ for $u \in \check{U}_t$. Over time, the placement is incrementally improved according to the client movement situation.

C. Online Algorithm Design

The online algorithm proposed above could achieve reasonable performance when the mobility of clients is limited (as shown in Figure 9). However, it can be expected that over time the performance loss will accumulate and finally we will reach a point where paying the penalty for reconfiguring the system (i.e., possibly migrating the SEs for static clients) outperforms keeping the incremental updates done by INCU. Now, we analyze when such a point will be reached, based on which we devise a more performant online algorithm.

We use ITEM(t) to denote the total cost produced by the ITEM algorithm and INCU(t) to denote the total cost produced by INCU given the instance at time t. The performance loss of INCU is defined as $e(t) = |\text{INCU}(t) - \text{ITEM}(t)|/\text{ITEM}(t)$. The accumulative performance loss is given by $h(t) = \sum_{i=1}^{t} e(t) = h(t-1) + e(t)$. We assume that $h(t)$ has been upper-bounded by a constant factor $\theta > 0$, meaning that it is very undesirable that the accumulated performance loss $h(t)$ is beyond the threshold θ, reflecting the stringent performance requirement set possibly by a service level agreement (SLA). We say that the system is stable when $h(t) \leq \theta$ and unstable otherwise. In the unstable case, a large penalty ω ($\omega > 0$) has to be paid compensating for the performance constraint violation. Our goal is to delay the application of ITEM as much as possible, while avoiding such violations. The rationale behind this is that applying ITEM is costly as some of the SEs need to be migrated incurring interruptions to static clients as well as additional network cost. To this end, we define the following reward function

$$r(t) = \begin{cases} \phi t, & h(t) \leq \theta, \\ -\omega, & h(t) > \theta. \end{cases} \tag{40}$$

This reward function represents that if the system is stable (and INCU is applied) we receive the reward of ϕt ($\phi > 0$) that is proportional to the elapsed time during which the system stays stable; we pay the penalty of ω otherwise. Our problem becomes to decide when to apply ITEM such that the expected reward is maximized. We notice that this problem actually corresponds to a case of the optimal stopping problem. Inspired by this observation and following the results from optimal stopping theory [44], we have the following theorem.

Theorem 6: The expected reward is maximized when ITEM is applied at time t such that it satisfies

$$\phi t \geq \frac{(\phi + \omega) F_e(\theta - h(t)) - \omega}{1 - F_e(\theta - h(t))}, \tag{41}$$

where $F_e(\cdot)$ is the cumulative distribution function of $e(t)$.

Proof: Note that the mobility is assumed to be a random variable and therefore, $e(t)$ and $h(t)$ are all random variables. We denote by $P(h(t) \leq \theta)$ the probability that $h(t) \leq \theta$. We have the expected reward at time t as

$$\begin{aligned} \mathbb{E}(r(t)) &= \phi t P(h(t) \leq \theta) - \omega(1 - P(h(t) \leq \theta)) \\ &= (\phi t + \omega) P(h(t) \leq \theta) - \omega. \end{aligned} \tag{42}$$

Given the series of performance loss observations $e(1), \ldots, e(t)$ denoted as $\vec{e}(t)$, the expected reward at

Algorithm 4 OPTS – Optimal Stopping

```
1: $t \leftarrow 1$;
2: while true do
3:     $e(t) \leftarrow |\text{INCU(t)} - \text{ITEM}(t)|/\text{ITEM}$;
4:     $h(t) \leftarrow h(t-1) + e(t)$;
5:     $F_e(\cdot) \leftarrow \text{KDE}(\vec{e}(t))$;                ▷ Obtain the CDF
6:     if $F_e(\theta - h(t)) > (\phi t + \omega)/(\phi(t+1) + \omega)$ then
7:         return INCU($t$);                ▷ Incremental update
8:     else
9:         return ITEM($t$);                ▷ Complete update
10:    $t \leftarrow t + 1$;
```

time $t+1$ is given by

$$\begin{aligned} &\mathbb{E}[r(t+1) \mid \vec{e}(t)] \\ &= (\phi t + \phi + \omega)P(h(t+1) \leq \theta) - \omega \\ &= (\phi t + \phi + \omega)P(h(t) + e(t+1) \leq \theta \mid \vec{e}(t)) - \omega \\ &= (\phi t + \phi + \omega)P(e(t+1) \leq \theta - h(t) \mid \vec{e}(t)) - \omega \\ &= (\phi t + \phi + \omega)F_e(\theta - h(t)) - \omega. \end{aligned} \tag{43}$$

Assume at time t the expected reward is maximized and we apply ITEM at the beginning of time slot t. It follows that $\mathbb{E}(r(t+1) \mid \vec{e}(t)) \leq r(t) = \phi t$. Therefore, we have $\mathbb{E}[r(t+1) \mid \vec{e}(t)] = (\phi t + \phi + \omega)F_e(\theta - h(t)) - \omega \leq \phi t$, from which we derive $\phi t \geq \frac{(\phi+\omega)F_e(\theta - h(t)) - \omega}{1 - F_e(\theta - h(t))}$. □

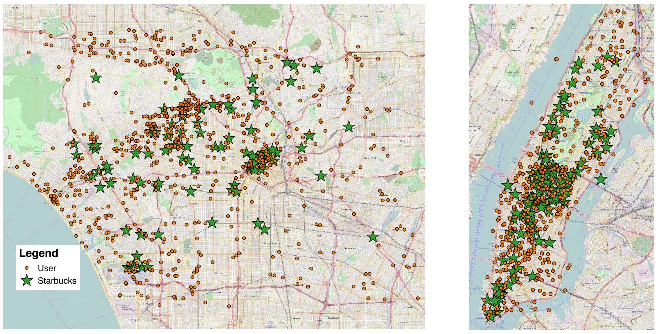
The above theorem provides us the insight for an online algorithm design. The pseudocode of the algorithm (namely OPTimal Stopping, OPTS) is listed in Algorithm 4. First, we apply ITEM for an initial optimized SE placement. Hereafter in every time slot, we check whether Equation (41) can be satisfied (line 6). If so, we apply ITEM again to reconfigure the system; otherwise, we simply apply INCU to carry out incremental updates. The cumulative distribution function $F_e(\cdot)$ can be obtained by smoothing the empirical cumulative distribution function using kernel density estimation (line 5).

The time complexity of OPTS is given by $O(\Delta(m^2(m + n) + n^2))$. As we can see in Algorithm 4, the most time consuming components are KDE, INCU, and ITEM. KDE takes a constant amount of time if we keep a limited number of history records. INCU applies the same logic as ITEM, but with a reduced set of users. Thus, its time complexity is upper bounded by that of ITEM. Overall, the time complexity of OPTS is dominated by ITEM.

The deployment of ITEM and OPTS can be done on top of the edge management frameworks such as KubeEdge[2] within an geographic region like a city. For example, the algorithm will be run at the EdgeController component of KubeEdge. The EdgeController obtains the necessary parameters required by our algorithms by monitoring the edge environment with the MetaManager component.

V. Evaluation

With real-world data we validate the performance of ITEM and OPTS. All the experiments were conducted on a Linux server with an Intel Core i9-9940X CPU and 64GB DRAM.



(a) Los Angeles (b) New York City

Fig. 5. Location of Starbucks shops (i.e., location for envisioned edge nodes) and distribution of Twitter clients in the selected two cities.

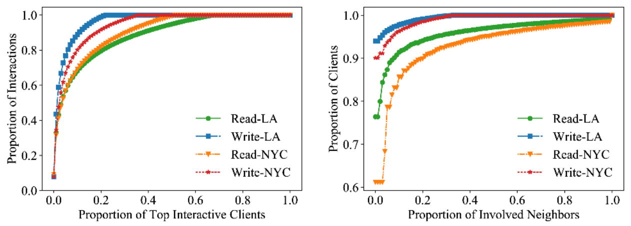
A. Offline Performance

1) Dataset: We focus on social VR as it is representative for collaborative edge applications. Unfortunately, there is no available dataset for social VR applications which we can use directly. To overcome this limitation, we follow the "workload composition" method proposed in [45] and decide to use multiple datasets with relevant attributes to compose a reasonable dataset for our target application. In particular, we obtain a social network dataset by crawling the Twitter website to emulate the user interactions in social VR. Our dataset contains a Twitter social graph as well as client locations in GPS coordinates. We select two major cities in the U.S., namely Los Angeles and New York City, and we prune the dataset keeping the clients from the two cities. The two cities have quite different client distributions which is more uniform in Los Angeles and is more concentrated in New York City (see Figure 5). For the locations of envisioned edge nodes, we decide to use the locations of the Starbucks due to the fact that Starbucks shops in a city usually provide a decent coverage for the clients. In addition, the distribution of Starbucks shops actually follows the population density, making them perfect locations for edge nodes deployment in the future.

The dataset pruned for Los Angeles (denoted by Twitter-LA) contains 7553 clients in total. We keep the relationship of the clients as is in the Twitter social graph. Although the frequency of client interaction is not available with the used dataset, we can actually synthesize it following the real-world distributions revealed by [46]. The CDFs of the synthetically generated frequencies of client interaction (including both "read" and "write") are depicted in Figure 6a. We obtain the locations of all the Starbucks shops in the city of Los Angeles and there are in total 105 shops considered, as shown in Figure 5a. We assume that each Starbucks shop will be equipped with an edge node that is attached to its AP. The network delay between any two Starbucks APs is measured by their geographical distance. We assume that each client is in the vicinity of the Starbucks shop that is closest to her.

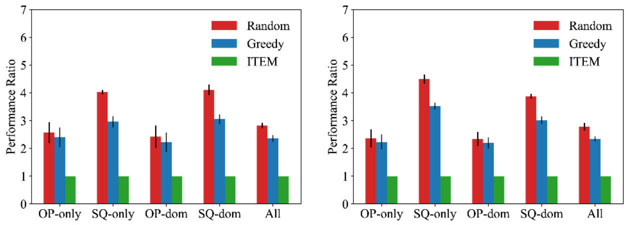
The dataset pruned for New York City (denoted by Twitter-NYC) contains 6068 clients and we synthesize the frequency of client interaction following the same procedure as above (see Figure 6(b)). The number of considered Starbucks shops in New York is 117 in total, as illustrated in Figure 5(b).

[2] KubeEdge edge computing framework: https://kubeedge.io/en/



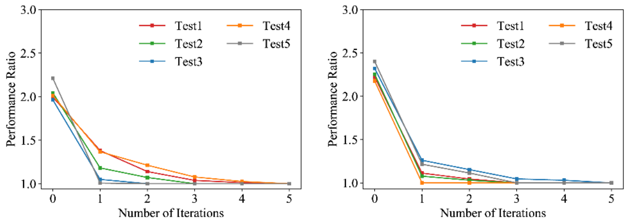
(a) CDF of client interaction (b) CDF of involved neighbors

Fig. 6. Distribution of client interaction (i.e., read and write) in the synthesized data. (a) shows that the $y \times 100\%$ of interactions are from $x \times 100\%$ most interactive clients. (b) shows that $y \times 100\%$ of clients involve at most $x \times 100\%$ of her neighbors in her interactions.



(a) Los Angeles (b) New York City

Fig. 7. Performance of ITEM under varying cost conditions.



(a) Los Angeles (b) New York City

Fig. 8. Convergence of ITEM on different datasets.

2) Settings: Our implementation of ITEM is based on a max-flow implementation detailed in [47]. The activation cost for each of the edge nodes is generated randomly following a uniform distribution to incorporate the heterogeneity of edge nodes with respect to both the hardware specification and physical location. For the placement cost, we assume that there are three different price levels and the ratio between the base prices of adjacent levels is set to 2. This is to represent the heterogeneity in the cost of using an edge node in different areas in the city. The actual placement price for each client placing her SE on an edge node is generated following a normal distribution with the average base and standard deviation 0.5×base. This is inspired by the Gaussian distribution of hourly electricity price in some US cities [48]. We propose to mimic the same trend for edge resource prices since both electricity and edge resources are limited shared resources among a large population. The communication frequency f_u of each client is set to be the sum of frequency of access originated from client u. For the co-location cost, we randomly generate the coefficients following a uniform distribution. We first compare with two baseline solutions of interest: Random (randomly generated placement) and Greedy (the greedy placement where SEs are placed with closest proximity). We also choose the theoretical optimum as our baseline. However, due to the large scale of the dataset, we were not able to obtain the values for the optimum in reasonable time. We will report the comparison against the theoretical optimum in the next experiment where we use a smaller-scale dataset. While the cost configuration is out of the scope of this paper, we conduct the comparison under a variety of cost conditions by tuning the parameters in the cost model: (1) Only the operation costs (OP-only) where the parameters in E_D and E_C are set to zero; (2) Only the service quality costs (SQ-only) where the parameters in E_A and E_B are set to zero; (2) The operation costs dominate (OP-dom) where we set the operation costs at roughly the level of 10× of the service quality costs; (3) The service quality costs dominate (SQ-dom) where we set the service quality costs at roughly the level of 10× of the operation costs; (4) All costs are significant (All) where operations costs and service quality costs are set at roughly the same level.

3) Results: The performance of ITEM, compared with Random and Greedy, is validated in Figure 7. All the results are obtained by averaging five independent runs and are normalized to the results of ITEM. As we can see that, ITEM outperforms both Random and Greedy and the gain achieved by ITEM is more than 2× in general under all considered cost conditions in both the Twitter-LA and Twitter-NYC datasets. This means that ITEM generalizes well and can handle most realistic scenarios. Figure 8 shows the converging speed of ITEM in both the Twitter-LA and the Twitter-NYC scenarios with five independent runs each. In general, ITEM converges very fast; it reduces the cost significantly in the first few iterations and converges within five iterations in most cases.

B. Online Performance

1) Dataset and Settings: For the online case, mobility is of concern. We adopt the same "workload composition" technique as discussed before. We use the Rome Taxi dataset and synthesize social networks with fitted client interactions. The time granularity is set to one minute since this is the finest time granularity we could use to derive a complete set of data points from the used dataset. The driving speed limit in Rome (urban area) is 50 km/h which translates into around 830 meters per minute. That is, during one minute, the movement of the user is below the typical communication range of an access point in networks like 4G and even 5G [49]. We choose a 6-hour period (15h to 20h inclusive on date Feb 12, 2014) from the dataset. The number of clients varies from hour to hour but is generally around 300 during the considered time period. The clients are moving around the city over time and the time granularity is set to one minute. We simulate a social graph on these clients following a power-law distribution with exponent 2.5 and we generate the frequency of client interaction using the same approach as for Twitter-LA. We envision an edge computing system with 15 edge nodes that are deployed in the city of Rome and the locations of the edge nodes are chosen from the major metro stations in Rome [25].

We implement a discrete-time simulator where at the beginning of each time slot (e.g., each minute here) we obtain the set of clients that have moved and then, we invoke the ITEM algorithm on those clients to obtain new placement decisions.

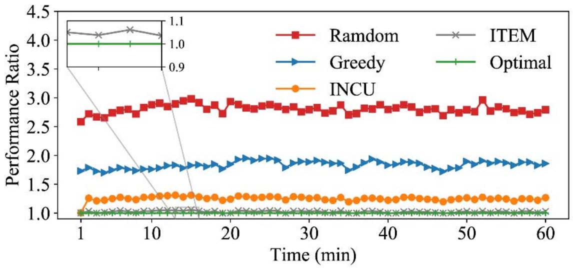
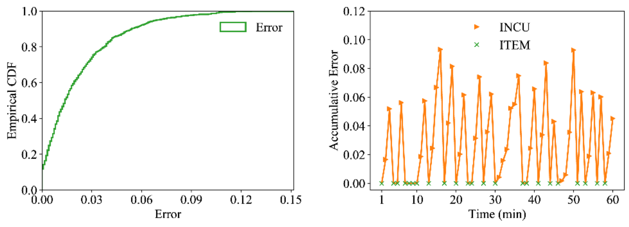


Fig. 9. Performance comparison of online service placement solutions.



(a) CDF

(b) Accumulative error

Fig. 10. (a) Empirical CDF obtained by the KDE method. (b) Performance of OPTS with either INCU and ITEM applied in each decision round.

We choose parameters that are generated following the same settings as in the offline case and compare the results with that of Random, Greedy, and Optimal (theoretical optimal results).

2) Performance Comparison: Figure 9 depicts the results for online performance evaluation. The experiments are done independently for 6 hours as described in the settings. We only show the hour of 16h as the tests in other hours show similar behavior. All the values in the plots are normalized to the theoretical optimum and are averaged among five independent runs. As we can see both INCU outperforms both Random and Greedy as expected and can achieve an overall cost reduction of around 130% and 50% under any circumstances in the simulated scenario. ITEM performs very closely to the theoretical optimum – the gap is within 5% (see the magnified area in Figure 9). The performance of the algorithms are stable over all the time slots we consider in this experiment.

3) Performance of OPTS: We take the dataset of 16h and illustrate the accumulative error incurred gradually through the 60 mins. The empirical CDF of the error obtained by the KDE method is depicted in Figure 10(a). We set the threshold $\theta = 10\%$, meaning that a large penalty (set as approximately two times the normal average total cost) is paid when the error exceeds this threshold. Figure 10(b) shows the performance of OPTS. With incremental updates by INCU the error accumulates gradually. Overall, OPTS does a great job in controlling the risk of threshold violation. OPTS always makes the re-balance decision when the error is close to the threshold. On the other hand, it never makes the re-balance decision too late such that the error exceeds the threshold. This confirms that the *OPTS can achieve the best while respecting performance requirement constraints*. However, OPTS only invokes ITEM for around 20 times in the whole 60 mins, resulting in over 67% reduction on the need of full updates.

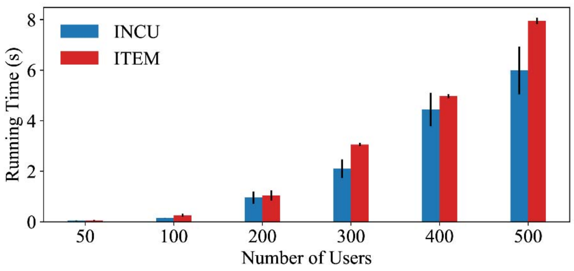


Fig. 11. Running time comparison between INCU and ITEM.

C. Running Time

We also measure the running time and the results are shown in Figure 11. We make a comparison between ITEM and INCU. In this experiment, we use the same edge node setting as in the last experiment and we synthesize the social network with various numbers of clients. As we can observe, when the number of clients is small, ITEM and INCU finish in a similar amount of time. With the increase of the number of clients, ITEM increases much faster than INCU and the gap between INCU and ITEM becomes quite significant. Given that the time slot for SE placement is usually at the scale of minutes, both algorithms should be practical.

VI. Related Work

A. Data Placement for Online Social Networks

Much work has been carried out on optimizing cost or performance via data placement or replication for online social networks [28]–[31]. Jiao *et al.* propose a cost-effective data placement policy that can guarantee QoS in online social networks [28]. They also investigate the problem by taking into account the social interactions among users [30]. Yu and Pan propose an associated data placement scheme to improve the colocation of associated data and localized data serving [29]. Zhou *et al.* explore the joint placement and replication of social network data with the goal of minimizing network traffic [31].

B. Resource Management in Edge Computing

In the presence of multiple edge clouds, resource management is of high importance as it directly dictates service quality and system efficiency. Research efforts have been made mostly on resource allocation and job scheduling [18]–[25]. Jia *et al.* study the load balancing among multiple edge clouds in [19]. Mukherjee *et al.* explore to reduce power and latency in multi-cloudlet environments via an optimal cloudlet selection strategy [20]. Tong *et al.* discuss workload placement for delay minimization in a hierarchical edge computing architecture [21]. Wang *et al.* [22] and Urgaonkar *et al.* [23] focus on stochastic frameworks for optimizing dynamic workload migration based on Markov Decision Processes (MDPs) and Lyapunov optimization techniques. Tan *et al.* study online job dispatching and scheduling in edge clouds [24]. Wang *et al.* propose an online mobility-oblivious resource allocation algorithm for edge computing [25]. Most recently, we have seen significant work on resource allocation, service placement, and request routing [50]–[54].

None of the existing models are able to characterize the joint impact of user interactions in collaborative edge applications and resource contention in edge nodes for service placement, which is captured in our model. Moreover, we incorporate the economic effects on activating and using edge nodes.

VII. Conclusion

In this paper, we conducted a formal study of the service placement problem for collaborative edge applications in a distributed edge network. We characterize the major challenges with a comprehensive cost model and propose a novel algorithm based on iteratively solving a series of minimum graph cuts. We also extend the algorithm to support the online case based on the optimal stopping theory. The performance of the proposed algorithms is confirmed by extensive experiments. As edge computing is picking up and more applications are developed for the edge environment, the solution provided in this paper will serve as a baseline and will foster future exploration in this direction.

Acknowledgment

The authors would like to thank the anonymous reviewers and the editor for their valuable comments and suggestions. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defense or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] L. Wang, L. Jiao, T. He, J. Li, and M. Muhlhauser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 468–476.

[2] *Mobile Edge Computing (MEC); Service Scenarios*, ETSI MEC-IEG, Sophia Antipolis, France, 2015.

[3] M. Mühlhäuser *et al.*, "Street lamps as a platform," *Commun. ACM*, vol. 63, no. 6, pp. 75–83, May 2020.

[4] J. Gedeon *et al.*, "From cell towers to smart street lamps: Placing cloudlets on existing urban infrastructures," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 187–202.

[5] Z. Hao, S. Yi, and Q. Li, "Edgecons: Achieving efficient consensus in edge computing networks," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–7.

[6] H. Gupta and U. Ramachandran, "FogStore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access," in *Proc. 12th ACM Int. Conf. Distrib. Event-based Syst.*, Jun. 2018, pp. 148–159.

[7] *Facebook Horizon*. Accessed: Sep. 26, 2020. [Online]. Available: https://www.oculus.com/facebook-horizon/

[8] D. Kumar, A. A. Ramkumar, R. Sindhu, and A. Chandra, "DeCaf: Iterative collaborative processing over the edge," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2019, pp. 1–9.

[9] S. Lu, Y. Yao, and W. Shi, "Collaborative learning on the edges: A case study on connected vehicles," in *Proc. 2nd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2019, pp. 1–8.

[10] A. Trivedi, L. Wang, A. Iosup, and H. Bal, "Sharing and caring of data at the edge," in *Proc. 3rd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2020, pp. 1–12.

[11] Q.-V. Pham *et al.*, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[12] F. Bonomi, R. A. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[13] *Open Edge Computing*. Accessed: Sep. 26, 2020. [Online]. Available: http://openedgecomputing.org

[14] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[15] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2019, pp. 1–16.

[16] 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. (SEC)*, Oct. 2016, pp. 155–167.

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

[18] 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 (ICNP)*, Nov. 2016, pp. 1–6.

[19] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[20] 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. 2019.

[21] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

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

[23] 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, Sep. 2015.

[24] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.

[25] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1281–1290.

[26] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Muhlhauser, "MOERA: Mobility-agnostic online resource allocation for edge computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1843–1856, Aug. 2019.

[27] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. M. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 689–702, Jun. 2015.

[28] L. Jiao, J. Li, T. Xu, and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2012, pp. 1–10.

[29] B. Yu and J. Pan, "Location-aware associated data placement for geo-distributed data-intensive applications," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 603–611.

[30] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 28–36.

[31] J. Zhou and J. Fan, "JPR: Exploring joint partitioning and replication for traffic minimization in online social networks," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1147–1156.

[32] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," in *Proc. 30th Annu. ACM SIGACT-SIGOPS Symp. Princ. Distrib. Comput. (PODC)*, 2011, pp. 267–276.

[33] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[34] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 2566–2574.

[35] F. Khan. (2015). *The Cost of Latency*. [Online]. Available: https://www.digitalrealty.com/blog/the-cost-of-latency

[36] J. Saldana and M. Suznjevic, *Traffic of Online Games*, document IETF 87, 2013.

[37] S.-H. Lim, J.-S. Huh, Y. Kim, G. M. Shipman, and C. R. Das, "D-factor: A quantitative model of application slow-down in multi-resource shared systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 271–282, Jun. 2012.

[38] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: Challenges and approaches," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 3, pp. 55–60, Jan. 2010.

[39] J. Krarup and P. M. Pruzan, “The simple plant location problem: Survey and synthesis,” *Eur. J. Oper. Res.*, vol. 12, no. 1, pp. 36–81, Jan. 1983.

[40] J. B. Orlin, “Max flows in O(nm) time, or better,” in *Proc. 45th Annu. ACM Symp. Symp. Theory Comput. (STOC)*, 2013, pp. 765–774.

[41] V. King, S. Rao, and R. Tarjan, “A faster deterministic maximum flow algorithm,” *J. Algorithms*, vol. 17, no. 3, pp. 447–474, Nov. 1994.

[42] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, “Fast approximate energy minimization with label costs,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1–8.

[43] P. Zhang, “A new approximation algorithm for the unbalanced min s–t cut problem,” *Theor. Comput. Sci.*, vol. 609, pp. 658–665, Jan. 2016.

[44] Y. S. Chow, H. Robbins, and D. Siegmund, *The Theory of Optimal Stopping*. Boston, MA, USA: Houghton Mifflin, 1991.

[45] O. Kolosov, G. Yadgar, S. Maheshwari, and E. Soljanin, “Benchmarking in the dark: On the absence of comprehensive edge datasets,” in *Proc. 3rd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2020, pp. 1–11.

[46] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao, “User interactions in social networks and their implications,” in *Proc. 4th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2009, pp. 205–218.

[47] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[48] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, “Cutting the electric bill for Internet-scale systems,” in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, 2009, pp. 123–134.

[49] M. Shafi *et al.*, “5G: A tutorial overview of standards, trials, challenges, deployment, and practice,” *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

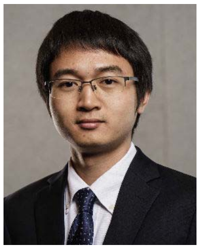
[50] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, “Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug. 2018.

[51] 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. (SECON)*, Jun. 2018, pp. 1–9.

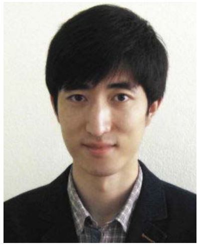
[52] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, “It’s hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources,” in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 365–375.

[53] V. Farhadi *et al.*, “Service placement and request scheduling for data-intensive applications in edge clouds,” in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 1278–1287.

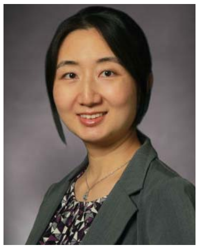
[54] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, “Joint service placement and request routing in multi-cell mobile edge computing networks,” in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 10–18.



Lin Wang received the Ph.D. degree (Hons.) in computer science from the Institute of Computing Technology, Chinese Academy of Sciences. He has been a Visiting Researcher with the IMDEA Networks Institute, Spain, from 2012 to 2014, a Research Associate with SnT Luxembourg from 2015 to 2016, and a Group Leader with TU Darmstadt, Germany, from 2016 to 2018. He is currently an Assistant Professor with VU Amsterdam, The Netherlands, and an Adjunct Professor with TU Darmstadt. His current research interests include edge computing, edge storage, intermittent computing, and in-network computing. He received the Athene Young Investigator Award from TU Darmstadt in 2018 and has been a PI of the Collaborative Research Center MAKI funded by the German Science Foundation (DFG).



Lei Jiao (Member, IEEE) received the Ph.D. degree in computer science from the University of Göttingen, Germany. He worked as a member of Technical Staff with Nokia Bell Labs, Dublin, Ireland, and also as a Researcher with IBM Research, Beijing, China. He is currently an Assistant Professor with the Department of Computer and Information Science, University of Oregon, Eugene, OR, USA. His research interests include exploring optimization, control, learning, mechanism design, and game theory to manage and orchestrate large-scale distributed computing and communication infrastructures, services, and applications. He has published articles in journals, such as JSAC, TON, TMC, and TPDS, and in conferences, such as MOBIHOC, INFOCOM, ICNP, ICDCS, SECON, and IPDPS. He served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) Series on Network Softwarization and Enablers. He was on the program committees of many conferences, including MOBIHOC, INFOCOM, ICDCS, and IWQoS, and was the Program Chair of multiple workshops with INFOCOM and ICDCS. He was also a recipient of the Best Paper awards of IEEE CNS 2019 and IEEE LANMAN 2013, and the 2016 Alcatel-Lucent Bell Labs U.K., and the Ireland Recognition Award.



Ting He (Senior Member, IEEE) received the B.S. degree in computer science from Peking University, China, in 2003, and the Ph.D. degree in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2007. She is currently an Associate Professor with the School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA, USA. Her research interests include in the broad areas of computer networking, network modeling and optimization, and statistical inference. She received the Research Division Award and multiple Outstanding Contributor awards from IBM, the Most Collaboratively Complete Publications Award from ITA, the Best Paper Award at ICDCS 2013, the Outstanding Student Paper Award at ACM SIGMETRICS 2015, and the Best Student Paper Award at ICASSP 2005. She was the Membership Co-Chair of the ACM N2Women from 2013 to 2014 and was listed in N2Women: Rising Stars in Networking and Communications in 2017. She has been an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS since 2017 and the IEEE/ACM TRANSACTIONS ON NETWORKING since 2017, and the Area TPC Chair of the IEEE INFOCOM since 2021.

Jun Li (Senior Member, IEEE) received the B.S. degree from Peking University in 1992, the M.E. degree from the Chinese Academy of Sciences in 1995, and the Ph.D. degree (D.Phil., Hons.) from UCLA in 2002, all in computer science. He is currently a Professor with the Department of Computer and Information Science and the Founding Director of the Center for Cyber Security and Privacy, University of Oregon. His research is focused on networking, distributed systems, and network security, with more than100 peer-reviewed publications. His current research interests include Internet routing and architecture, software-defined networking, social networking, cloud computing, the Internet of Things, blockchain, cryptocurrency, and various security topics in these areas. He is an NSF CAREER Awardee in 2007. He received the Presidential Scholarship for his M.E. degree. He has served on U.S. National Science Foundation research panels and more than 70 international technical program committees, including chairing six of them. He also serves on the Editorial Board of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and a few conference or workshop steering committees. He is a Senior Member of the ACM.

Henri Bal received the M.Sc. degree in mathematics from the Delft University of Technology in 1982 and the Ph.D. degree in computer science from Vrije Universiteit, Amsterdam, in 1989. He studies parallel and distributed programming systems in combination with real-world applications. His group produced programming environments, such as the Orca language, MagPIe, Ibis, Satin, JavaGAT, and SWAN. He is currently a Full Professor and the heads a research group on High-Performance Distributed Computing, Vrije Universiteit. He is the winner of the Euro-Par 2014 Achievement Award, a member of the Informatics Section of the Academia Europaea, the Scientific Director of the ASCI Research School, and a Coordinator of the DAS infrastructure. He is past Program Chair of the HPDC and CCGrid conferences and an author of three books, including Modern Compiler Design.