Service Entity Placement for Social Virtual Reality Applications in Edge Computing

Lin Wang
TU Darmstadt
Darmstadt, Germany
wang@tk.tu-darmstadt.de

Lei Jiao
University of Oregon
Eugene, OR, USA
jiao@cs.uoregon.edu

Ting He
Pennsylvania State University
State College, PA, USA
t.he@cse.psu.edu

Jun Li
University of Oregon
Eugene, OR, USA
lijun@cs.uoregon.edu

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

***Abstract*—While social Virtual Reality (VR) applications such as Facebook Spaces are becoming popular, they are not compatible with classic mobile- or cloud-based solutions due to their processing of tremendous data and exchange of delay-sensitive metadata. Edge computing may fulfill these demands better, but it is still an open problem to deploy social VR applications in an edge infrastructure while supporting economic operations of the edge clouds and satisfactory quality-of-service for the users.**

This paper presents the first formal study of this problem. We model and formulate a combinatorial optimization problem that captures all intertwined goals. We propose ITEM, an iterative algorithm with fast and big "moves" where in each iteration, we construct a graph to encode all the costs and convert the cost optimization into a graph cut problem. By obtaining the minimum s-t cut via existing max-flow algorithms, we can simultaneously determine the placement of multiple service entities, and thus, the original problem can be addressed by solving a series of graph cuts. Our evaluations with large-scale, real-world data traces demonstrate that ITEM converges fast and outperforms baseline approaches by more than $2\times$ in one-shot placement and around $1.3\times$ in dynamic, online scenarios where users move arbitrarily in the system.

I. Introduction

Virtual Reality (VR) and Augmented Reality (AR) are commonly believed to be the killer applications for the emerging edge computing paradigm [1], where resources are available for service provisioning in much closer proximity to end users with low-delay and low-jitter network connections. VR fits well in this paradigm due to its resource-hungry and delay-sensitive nature. As an inevitable technology trend, VR has gained popularity. Recently at WWDC 2017, Apple released a new tool ARKit for developing VR applications on its platforms. In the meanwhile, Facebook is augmenting its social network with Virtual/Augmented Reality and recently, it released its new social network platform named Spaces.

In fact, edge computing has already been adopted for VR in industry for handling the situation posed by resource limitation of local processing and large, unpredictable latency of remote cloud-based processing. For example, Apple uses iMac to support VR rendering and Facebook Oculus employs a Windows PC for computing. According to the Open Edge Computing initiative [2], an edge cloud will be able to offer compute and storage resources to any user in close proximity through an open and globally standardized mechanism, which allows a set of edge clouds in the same geographical region to form a shared edge resource pool. With the help of lightweight virtualization techniques, resources will be allocated at a fine granularity (e.g., per user) subject to quality-of-service (QoS) requirements and system-wide optimization goals.

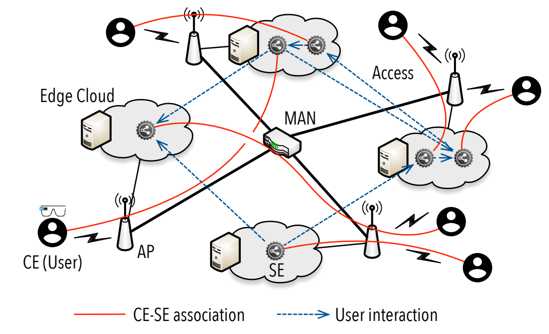


Fig. 1. An example scenario to illustrate the service entity placement problem for social VR applications in the edge environment. (AP: Access Point, CE: Client Entity, SE: Service Entity)

In this paper, we focus on VR applications with online social network support (referred to as social VR applications) and we study the problem of placing service entities for social VR applications in the edge environment. As illustrated in Figure 1, a social VR application consists of two parts: service entity (SE) and client entity (CE, on user device). The SE, defined as the bundle of the user's personal data and the processing logics on the data, takes care of the user state and computation-intensive tasks such as scene rendering, object recognition and tracking, and the interaction between users, while the CE is only in charge of displaying the video frames rendered by the SE and monitoring user behaviors. The service entity placement problem is to decide where to place the SE of each user among the edge clouds in order to achieve economic operations of the edge clouds as well as satisfactory QoS for the users. We call this problem *edge service entity placement (ESEP)*. While motivated by social VR applications, the ESEP problem is fundamental for any applications that require interactions both between each user and its service and between (the services of) different users.

The ESEP problem is non-trivial mainly due to the following intertwined challenges: (1) Edge clouds are heterogeneous in terms of activation and running costs and thus, achieving the best economic outcome would *push the SEs to edge clouds with lower costs*; (2) SEs need to exchange metadata frequently with the associated CEs and other SEs, as users are updating their scenes and interacting with each other in social VR applications, e.g., accessing the state of other users or collaboratively completing a task, and thus, *placing SEs closer to each other would result in better QoS perceived by the users*; (3) Due to the fact that edge clouds are not intentionally designed to simultaneously accommodate a large number of SEs, especially for VR applications where specific hardware such as GPU may be involved, resource contention needs to be controlled, which suggests that *each edge cloud should not be too crowded with SEs*. The second challenge is unique to ESEP and all these intrinsically intertwined goals together complicate the problem.

Priori research efforts fall short of fully addressing the above challenges all together. On the one hand, most studies on edge computing are focused on computation offloading techniques and hardware/software architectures [3], [4]; edge resource management proposals generally lack the specific consideration of the interactions between users as in the case of social VR applications [5]–[11]. 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 etc. but neglect the impact of multiple important factors in the edge context such as activation cost and resource contention [12]–[16], which fundamentally change the settings and require different solution approaches. While general service placement problem has been extensively explored in various settings [17], [18], no existing algorithms are known to solve the ESEP problem.

Summary of contributions. We present the first formal study of ESEP and make the following contributions:

• We model the identified challenges with four types of cost including the activation cost, the placement cost, the proximity cost, and the colocation cost. The considered cost models are comprehensive yet practical, and are general enough to capture a wide range of concrete performance measures in reality. Based on these models, we formulate the ESEP problem as a combinatorial optimization, which we show is NP-hard.

• We propose the ITerative Expansion Moves (ITEM) algorithm to solve the formulated combinatorial optimization problem based on iteratively solving a series of minimum s-t cut instances. In each iteration, ITEM selects an edge cloud and performs an "expansion move", which is a local optimization of the current SE placement by determining whether each SE should stay as is or move to the selected edge cloud. We show that an expansion move is actually equivalent to solving a minimum s-t cut problem on a carefully constructed graph and we present the technique to build the graph with all of our costs encoded. ITEM has multiple advantages: (1) The graph construction is deterministic and easy to implement; (2) Placement decisions are made with big "moves", i.e., not only one but many SEs can be determined simultaneously in each iteration; (3) The decision-making is based on existing polynomial-time max-flow algorithms; (4) It converges fast, thanks to the "big move" in each iteration.

• We carry out extensive experiments with large-scale real-world data traces to validate the performance of ITEM. For the offline case, we employ a Twitter dataset with realistic user interactions and derive edge cloud locations from Starbucks locations. We select two major cities in the US, namely Los Angeles (LA) and New York City (NYC), to conduct our evaluations. For the LA scenario, we collect the locations of 105 Starbucks shops and generate a subset of the Twitter dataset with 7553 users by pruning the Twitter dataset according to user location. For the NYC scenario, the number of considered Starbucks shops is 117 and the pruned dataset contains 6068 users. The user interactions between users are synthesized following realistic distributions disclosed in [19]. For the online case, we use a Rome taxi dataset with around 300 taxi trajectories and we synthesize social interactions between the taxis. We select 15 major metro stations in Rome as edge cloud locations. Using these real-world datasets, we compare ITEM with baseline placement solutions and the key findings are: ITEM can well balance the four types of cost and outperforms the baselines up to $2\times$ and $1.3\times$ in offline and online cases, respectively, while exhibiting fast convergence.

TABLE I
LIST OF MAIN NOTATIONS

P	set of edge clouds
U	set of users
U_p	set of users with SEs placed on edge cloud p
p_u	edge cloud where the SE of user u is placed
p_u^*	edge cloud attached to the AP that user u is connected to
$\mathbf{p}$	set of SE placement for all the users, i.e., $\{p_1, ..., p_m\}$
$\mathbf{p}^q$	set of SE placement after expansion move on edge cloud q
f_u	comm. frequency between the CE and the SE of user u
$f_{u,v}$	frequency of access from user u to user v
$d(p, q)$	network delay between edge cloud p and q
m_p	number of SEs on edge cloud p
a_p	activation cost of edge cloud p
$l_u(p_u)$	cost of placing the SE for user u on edge cloud p_u
α_p, β_p	parameters in colocation cost of edge cloud p
L	set of all interacting user pairs
$\mathbf{x}$	binary decisions in an expansion move, i.e., $\{x_1, ..., x_m\}$

II. PROBLEM FORMULATION

We provide model for the system and introduce four types of cost, based on which we formulate the ESEP problem. Table I lists the main notations we will use throughout the paper.

A. The System

We consider a metropolitan-area edge computing system which consists of a set of n edge clouds, denoted by $P = \{p_1, \ldots, p_n\}$, that are dispersed in a city, e.g., inside bars or restaurants. Each edge cloud is accompanied by an access point (AP) that allows the user to connect to the platform. We assume that resources in the edge clouds are virtualized using container-based lightweight virtualization technologies and can thus be allocated and shared flexibly. All the edge clouds are connected to a metropolitan area network (MAN) inside a city and the delay between each pair of edge clouds p and q is given by function $d(p, q)$ for $p, q \in P$.

We consider the problem of provisioning a social VR application on the given set of edge clouds to serve m users distributed in the MAN. An exemplary scenario is illustrated

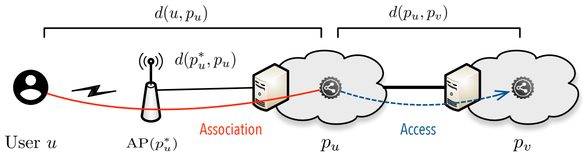


Fig. 2. Proximity cost includes the communication delay between the user and her SE and the interaction delay between users.

in Figure 1. The set of users is given by $U = \{u_1, \dots, u_m\}$. Each user u has access to the edge cloud system via its nearest AP, denoted by p_u^*, in her vicinity.[1] For each user $u \in U$, an SE is brought up on one of the edge clouds p_u to handle the computation within the application related to user u (e.g., rendering scenes, recognizing and tracking objects for user u). The SE encapsulates all the necessary runtime environment as well as the service state for the user. The frequency of communication between a user u and her SE p_u for updating scenes and monitoring user behaviors is denoted by f_u. In addition to employing the SE to process data, each user also interacts with other users within the application, e.g., accessing the profile or updates of other users or even completing a task collaboratively with other users. The frequency of access from user u to user v is given by $f_{u,v}$ ($f_{u,v} \triangleq 0$ if $u = v$). We denote by L the set of all interacting user pairs where $L = \{(u, v) \,|_{u,v \in U \wedge u \neq v}\}$.

B. Cost Models

We jointly consider multiple performance measures of the system by modeling four types of costs as detailed below.

Activation cost. For each of the edge clouds, if there is at least one SE being placed on it, then a static activation cost has to be paid. Such activation cost is incurred by the cooling or other maintenance efforts in the edge cloud irrespective of SEs. Assume that the set of SEs that are placed on edge cloud p is denoted by $U_p = \{u \,|_{p \in P \wedge p_u = p}\}$ and its cardinality is denoted by $m_p \in \mathbb{Z}^+$. Then, for each edge cloud $p \in P$ we define its activation cost by $a_p > 0$ if $m_p > 0$ and by 0 if $m_p = 0$, meaning that an edge cloud can be switched off if it hosts no SEs. The combined activation cost of all the edge clouds can be represented by

$$E_A = \textstyle\sum_{p \in P \wedge m_p > 0} a_p. \tag{1}$$

Placement cost. The placement cost is associated to the creation of SEs, which is incurred by the running and communication of the servers for the SEs. For each user $u \in U$, the cost of placing her SE onto edge cloud p is characterized by $l_u(p) > 0$, which can vary due to the location difference or the heterogeneity of the edge clouds. Denoting by $p_u \in P$ the placement decision for the SE of user u, the combined cost of placing all the SEs onto the set of edge clouds can then be represented by

$$E_L = \textstyle\sum_{u \in U} l_u(p_u). \tag{2}$$

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

Proximity cost. Proximity is a key performance metric in edge computing. The proximity measure of a user contains two parts: how timely a user (or her CE) interacts with her SE and how timely a user interacts with other users, as depicted in Figure 2. In order to give priority to frequent communication or interaction, we incorporate the communication frequency $f_u \geq 0$ between the CE and the SE of user u and the access frequency $f_{u,v} \geq 0$ from user u to user 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 user and its AP is actually irrelevant to the placement decision. For simplicity, we will omit it and will use the following form for the total proximity cost in the rest of the paper.

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

Colocation cost. The colocation cost is incurred by the resource contention among the SEs that are placed on the same edge cloud. This is considered unavoidable as edge clouds are not intentionally designed for large-scale resource multiplexing and performance isolation is typically difficult with light-weight virtualization. The performance degradation due to SE colocation can be characterized by $\alpha_p m_p^2 + \beta_p m_p$, where α_p and β_p are parameters. The reasoning behind this model is as follows. In the worse case, all the m_p SEs colocated on the same edge cloud rely on a single CPU core for processing and thus, the stretch on execution time, known as performance degradation ratio here, can be as large as m_p following the simple round-robin scheduling policy. Therefore, it is reasonable to use a general function $\alpha_p m_p + \beta_p$ to characterize the cost for the performance degradation of one of the colocated SEs given that performance degradation can also be induced by contention of resources other than CPU (e.g., memory, cache, or network) [20]. Summing up the costs for all the SEs on the same edge cloud, the total colocation cost for an edge cloud p can be represented by $m_p(\alpha_p m_p + \beta_p)$ as given above. The model is general in that the parameters α_p and β_p can be tuned to suit any practical needs depending on the edge cloud specification and the application type. Consequently, the total colocation cost in the system is characterized by

$$E_C = \textstyle\sum_{p \in P} (\alpha_p m_p^2 + \beta_p m_p). \tag{4}$$

The colocation cost also serves as a soft constraint for the maximum number of SEs that can be hosted by an edge cloud.

The above costs capture the system performance from different perspectives: activation and placement costs are from the operating expenses of the edge cloud provider, while proximity and colocation costs are from the perceived QoS of the users. Collectively, they provide a comprehensive model of the overall system performance.

C. Problem Formulation and Complexity

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

$$E = E_A + E_L + E_D + E_C. \tag{5}$$

Algorithm 1 ITEM

```
1: $flag \leftarrow true$;
2: while $flag = true$ do                        ▷ Search until convergence
3:     $flag \leftarrow false$;
4:     for $q \in P$ do                          ▷ Iterate over edge clouds
5:         $\mathbf{p} \leftarrow \{p_u \mid_{u \in U}\}$;            ▷ Current placement
6:         $\mathbf{p}^q \leftarrow$ EXPASION($\mathbf{p}, q$);        ▷ Expansion move
7:         if $E(\mathbf{p}^q) < E(\mathbf{p})$ then          ▷ Improvement found
8:             $\mathbf{p} \leftarrow \mathbf{p}^q$;
9:             $flag \leftarrow true$;
10: return $\mathbf{p} = \{p_u \mid_{u \in U}\}$;
```

Algorithm 2 Expansion

```
1: construct an auxiliary graph $G$ w.r.t. edge cloud $q$;
2: obtain the minimum $s$-$t$ cut on graph $G$ using Edmonds-Karp max-flow algorithm;
3: extract expansion decision $\mathbf{x}$ from the graph cut;
4: for $u \in U$ do                          ▷ Expanding edge cloud $q$
5:     $p_u \leftarrow q$ if $x_u = 1$;
6: return $\mathbf{p} = \{p_u \mid_{u \in U}\}$;
```

Note that different weights for different types of cost can be incorporated into the cost function directly, e.g., a_p for E_A, $l_u(p_u)$ for E_L, and α_p and β_p for E_C, respectively. Therefore, various tradeoff forms of the objective can be conveniently achieved. The goal of ESEP is to make decisions on the placement of SEs, i.e., determining p_u for all $u \in U$, for a social VR application, such that the total cost E of the edge computing system is minimized. Note that for simplicity the model does not impose hard constraints. This is reasonable for storage as it is generally considered cheap nowadays. Other hard constraints due to hardware (e.g., GPU) or security requirements can be easily modeled by restricting the placement to predetermined candidate set of edge clouds. In general, the problem is hard to solve and we show that

Theorem 1 (Complexity). *ESEP is NP-hard.*

Proof. We conduct the proof via a polynomial-time reduction from the uncapacitated facility location (UFL) problem, which is known to be NP-hard [21]. The reduction can be built by treating the distance and cost functions in a UFL instance as the placement and activation costs in an ESEP instance, respectively, and setting the other costs in ESEP to zero. □

III. Algorithm Design

Essentially, ESEP requires minimizing a nonlinear function in a 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 solved very efficiently, the searching complexity is thus significantly reduced.

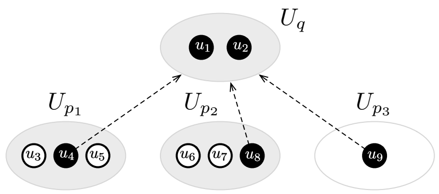


Fig. 3. An example to illustrate the expansion move, where edge cloud q is selected for expansion.

A. Iteractive Expansion Moves

We propose ITEM – an algorithm for the ESEP problem based on ITeractive Expansion Moves. The pseudocode of the algorithm is listed in Algorithm 1. An *expansion move* is an optimization process where an edge cloud $q \in P$ 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 cloud q, i.e., $p_u^q = q$. Let $\mathbf{p}^q = \{p_1^q, \ldots, p_m^q\}$ denote the placement after one feasible expansion on edge cloud 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 searching by applying expansion moves iteratively over all the edge clouds 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 10 iterations); see Figure 7(b).

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 minimum $E^q(\mathbf{x}^*)$, from which the new placement $\mathbf{p}^q$ will be produced. A simple example is illustrated in Figure 3, where users u_4, u_8 and u_9 switch to the selected edge cloud q from edge cloud p_1, p_2, and p_3, respectively, in the expansion move, while the other users stay at their current edge cloud. Users u_1 and u_2 will stay in edge cloud q according to the definition of expansion move. We have the following mappings. (Terms in circle represent that the decisions for the users that are already on q are always 1.)

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
$\mathbf{x}$	$\textcircled{1}$	$\textcircled{1}$	0	1	0	0	0	1	1

Obviously, 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.

B. *Transforming the Objective Function*

Given a current placement $\mathbf{p}$ and a selected edge cloud q, the objective of ESEP 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.

$$E_A^q(\mathbf{x}) = \sum_{\substack{p\in P\\ m_p^q>0}} a_p = \sum_{p\in P\setminus q} (a_p - a_p \prod_{u\in U_p} x_u) + \sigma_q, \quad (6)$$

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., $x_u = 0$. 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 cloud p. For a particular edge cloud $p \in P \setminus q$, the transformation is

$$-a_p \prod_{u\in U_p} x_u = \min_{z_p\in\{0,1\}} a_p((m_p-1)z_p - \sum_{u\in U_p} x_u z_p). \quad (7)$$

The final term σ_q in (6) is used to correct the case where edge cloud 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., $\sigma_q = a_q(1-\prod_{u\in U}\bar{x}_u)$.

The placement cost and the proximity cost are easy to be rewritten. Observe that

$$\begin{aligned} l_u(p_u^q) &= l_u(q)x_u + l_u(p_u)\bar{x}_u,\\ d(p_u^*, p_u^q) &= d(p_u^*, q)x_u + d(p_u^*, p_u)\bar{x}_u,\\ d(p_u^q, p_v^q) &= d(p_u, q)\bar{x}_u x_v + d(q, p_v)x_u\bar{x}_v + d(p_u, p_v)\bar{x}_u\bar{x}_v. \end{aligned}$$

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

$$\begin{aligned} E_L^q(\mathbf{x}) &= \sum_{u\in U}(l_u(q)x_u + l_u(p_u)\bar{x}_u),\\ E_D^q(\mathbf{x}) &= \sum_{u\in U} f_u(d(p_u^*, q)x_u + d(p_u^*, p_u)\bar{x}_u)\\ &+ \sum_{(u,v)\in L} f_{u,v}(d(p_u, q)\bar{x}_u x_v + d(q, p_v)x_u\bar{x}_v + d(p_u, p_v)\bar{x}_u\bar{x}_v). \end{aligned}$$

The colocation cost E_C^q after the expansion move is composed of two terms: the quadratic term and the linear term. Denoting by m_p^q the number of SEs being placed onto edge cloud p after the expansion move and L_p the current set of interacting user pairs that have their SEs on the edge cloud p, the quadratic term can be simplified as follows.

$$\begin{aligned} &\sum_{p\in P}\alpha_p(m_p^q)^2 = \alpha_q(\sum_{u\in U} x_u)^2 + \sum_{p\in P\setminus q}\alpha_p(\sum_{u\in U_p}\bar{x}_u)^2\\ &= \alpha_q(\sum_{u\in U} x_u)^2 + \sum_{p\in P\setminus q}\alpha_p(m_p^2 - 2m_p\sum_{u\in U_p} x_u + (\sum_{u\in U_p} x_u)^2)\\ &= \sum_{(u,v)\in L}\alpha_q x_u x_v + \sum_{p\in P\setminus q}(\sum_{(u,v)\in L_p}\alpha_p x_u x_v)\\ &\quad + \sum_{u\in U}\alpha_q x_u + \sum_{p\in P\setminus q}\alpha_p((1-2m_p)\sum_{u\in U_p} x_u + m_p^2). \quad (8) \end{aligned}$$

The above transformation is conducted by some algebra and by applying equation $x_u^2 = x_u$, which follows due to the fact that $x_u \in \{0,1\}$. We introduce an auxiliary $m \times m$ matrix $\mathbf{R}$ where

$$\mathbf{R}_{u,v} \triangleq \begin{cases} 1 & \text{if } p_u = p_v \neq q, \text{ for } u \neq v;\\ 0 & \text{otherwise.}\end{cases}$$

and an m-dimension indicator vector $\mathbf{y} = \{y_1, ..., y_m\}$ where $y_u = 1$ if $u \in U \setminus U_q$ and $y_u = 0$ otherwise. Note that the matrix $\mathbf{R}$ and vector $\mathbf{y}$ can be easily computed given the current placement $\mathbf{p}$ and the selected edge cloud q. The expression (8) can further be rewritten as

$$\begin{aligned} &\sum_{(u,v)\in L} -(\alpha_q + \alpha_{p_u}\mathbf{R}_{u,v})\bar{x}_u x_v \quad (9)\\ &+ \sum_{u\in U}((m+1)\alpha_q - y_u\alpha_{p_u}m_{p_u})x_u + \sum_{p\in P\setminus q}\alpha_p m_p^2. \end{aligned}$$

The linear term in the colocation cost can be rewritten as

$$\begin{aligned} \sum_{p\in P}\beta_p m_p^q &= \sum_{u\in U}\beta_q x_u + \sum_{p\in P\setminus q}\beta_p\sum_{u\in U_p}\bar{x}_u\\ &= \sum_{u\in U}(\beta_q - y_u\beta_{p_u})x_u + \sum_{p\in P\setminus q}\beta_p m_p, \quad (10) \end{aligned}$$

which is obviously linear in the x_u. In a nutshell, the colocation cost after the expansion move can be expressed as the sum of the two terms, i.e., $E_C^q = (9) + (10)$. Consequently, 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 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. We first define $\Delta(p_u, p_v, q) = d(p_u, q) + d(q, p_v) - d(p_u, p_v)$.

Graph construction. We now construct a graph G which encodes all the costs in our model. We first introduce m nodes, each of which corresponds to each user u. We then introduce n nodes to represent the set of edge clouds. Finally, we add a source node s and a sink node t, where s corresponds to decision 0 and t corresponds to decision 1. As a result, the set of nodes in G is given by $\{u\ |_{u\in U}\} \cup \{p\ |_{p\in P}\} \cup \{s, t\}$. To encode the activation cost, we first reparameterize the right hand of (6) such that each quadratic monomial has exactly one complemented variable (e.g., $\bar{x}z$) with nonnegative coefficient, i.e.,

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

For each $u \in U_p$, we add an edge from u to p_u with weight a_p. We also introduce an edge from each p to the sink t with weight a_p. This graph structure ensures that only weight of

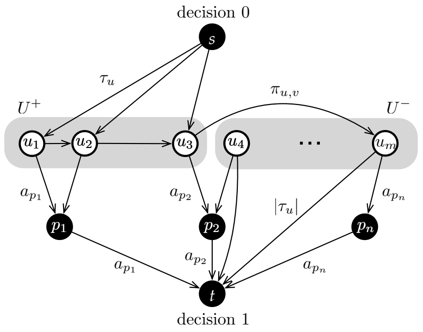


Fig. 4. Constructed graph G for encoding cost E^q.

a_p will be included in the graph cut. The encoding of the correction term σ_q in (6) is analogous to the above but we use a simpler test-and-reject technique to handle this term [22]. 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.

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

$$\sum_{(u,v)\in L} \pi_{u,v}\bar{x}_u x_v + \sum_{u\in U^+} \tau_u x_u + \sum_{u\in U^-} |\tau_u|\bar{x}_u + \kappa. \quad (11)$$

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

$$\pi_{u,v} = f_{u,v}\Delta(p_u, p_v, q) - (\alpha_q + \alpha_{p_u}\mathbf{R}_{u,v}),$$

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

$$\begin{aligned}\tau_u &= l_u(q) - l_u(p_u) + f_u(d(p_u^*, q) - d(p_u^*, p_u)) \\ &\quad + \sum_{v\in U}(f_{u,v}d(q,p_v) + f_{v,u}d(q,p_u) - f_{u,v}d(p_u,p_v)) \\ &\quad + (m+1)\alpha_q - y_u\alpha_{p_u}m_{p_u} + \beta_q - y_u\beta_{p_u},\end{aligned}$$

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

The placement of SEs now can be obtained by computing the minimum s-t cut on G by employing the Edmons-Karp max-flow algorithm. Specifically, we place the SEs after the expansion move according to the following rule.

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

Theorem 2 (**Correctness**). *Solving the minimization problem with objective E^q is equivalent to obtaining the minimum s-t cut of graph G as long as for any pair of interacting users $u, v \in U$ it satisfies $\Delta(p_u, p_v, q) \geq \alpha_q + \alpha_{p_u}\mathbf{R}_{u,v}/f_{u,v}$.*

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, this constraint follows in nature, while $\pi_{u,v} \geq 0$ is satisfied if we have $\Delta(p_u, p_v, q) \geq (\alpha_q + \alpha_{p_u}\mathbf{R}_{u,v})/f_{u,v}$.

Now we show the equivalence. For each edge cloud $p \neq q$, cost a_p is counted as long as there exists one edge 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 edge p-t will be included in the cut if there are more than one user that satisfies $x_u = 0$, contributing only a cost of a_p to E^q. For any pair of nodes 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 edge (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 edge s-u, meaning $x_u = 1$. Similarly, for any $u \in U^-$, the unary cost is counted if and only if the cut contains edge u-t, meaning that $x_u = 0$. □

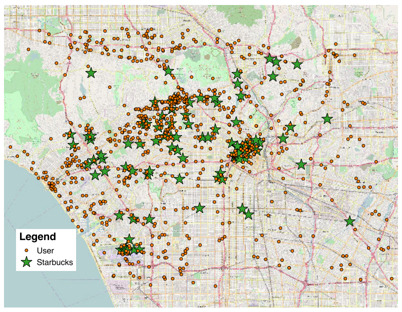
We can still construct a graph and solve the corresponding minimum s-t cut problem on the graph when Theorem 2 does not hold [23]. The graph construction process is explained in the following. Recall that in expression (11), for those edge cloud pairs u, v that satisfy Theorem 2, we have $\pi_{u,v} \geq 0$; for the others, we have $\pi_{u,v} < 0$. The problem is with those terms with $\pi_{u,v} < 0$ since edge weights on the graph should be nonnegative in order for the graph cut to be efficiently computed. To handle this situation, for the terms $\pi_{u,v}\bar{x}_u x_v$ with $\pi_{u,v} < 0$ we carry out the following transformation

$$\pi_{u,v}\bar{x}_u x_v = -\pi_{u,v}x_u(1 - \bar{x}_v) - \pi_{u,v}\bar{x}_v + \pi_{u,v}.$$

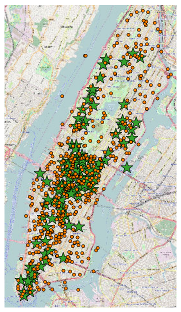
Note that after this transformation, we are able to incorporate the term $\pi_{u,v}\bar{x}_u x_v$ into the graph by introducing new nodes $\bar{u}$ which represent $\bar{x}_u$ and edges with weights $-\pi_{u,v} > 0$ between them, and edges from these nodes to node u with weight $-\pi_{u,v} > 0$. The constant term $\pi_{u,v}$ is merged into κ in expression (11). In every expansion move we solve the minimum s-t cut problem on this new graph and determine the placement of the SE for each user $u \in U$ according to the following rule:

$$p_u^q \triangleq \begin{cases} q & \text{if } x_u = 1 \text{ and } \bar{x}_u = 0, \\ p_u & \text{if } x_u = 0 \text{ and } \bar{x}_v = 1, \\ \text{undetermined} & \text{otherwise.}\end{cases}$$

For those undetermined SEs, the graph cut solution produces contradicting decisions on x_u and $\bar{x}_u$ and thus, it is not able to dictate their placement out from the two choices. Therefore, we carry out an extra procedure to place those undetermined SEs such that the total cost can further be reduced: for each undetermined SE, we set $p_u^q = q$ if placing it to q would bring cost reduction compared to keeping it at p_u.

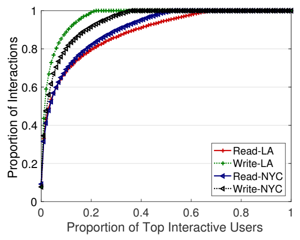


(a) Los Angeles

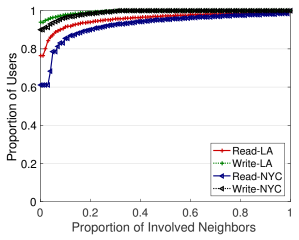


(b) New York City

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



(a) CDF of user interaction



(b) CDF of involved neighbors

Fig. 6. Distribution of user 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 users. (b) shows that $y \times 100\%$ of users involve at most $x \times 100\%$ of her neighbors in her interactions.

D. Discussion for Online Cases

So far, the ITEM algorithm works only for the static (offline) scenario. To handle online cases where placement decisions need to be updated following user movements, we assume a time-slotted system and we consider that a user has moved if the user switches from one AP to another. Note that the granularity of a time slot can be at the level of minutes due to the fact that users in social VR applications usually move at a walking speed and small movements will have no impact on the selection of APs, as a result of which no reoptimization is required. Keeping in mind that unexpected service interruption for unmoved users should be minimized, the design of the online algorithm is as follows: Denote the set of users that have moved as $\tilde{U}$ in each time slot. At the beginning of each time slot we first incorporate the migration overhead into the placement cost for users in $\tilde{U}$ according to the current locations of their SEs. Then, we carry out ITEM where we modify the expansion move by trimming the objective function E^q where we plug in the decisions $x_u = 0$ for all $u \in U \setminus \tilde{U}$. Finally, we place the SEs of users $u \in \tilde{U}$ according to the minimum graph cut solution; the placement for other users remain unchanged.

IV. Evaluation

With real-world data we validate the performance of ITEM for both offline and online cases.

A. Offline Performance

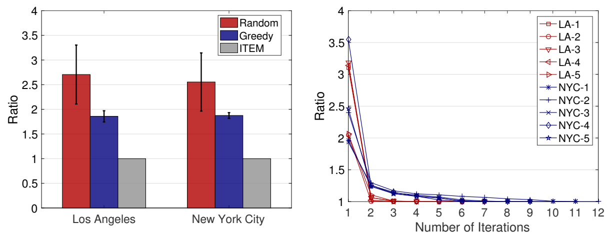
Dataset. We obtain a social network dataset by crawling Twitter website. The dataset contains a Twitter social graph as well as user locations in GPS coordinates. We select two major cities in the U.S., namely Los Angeles and New York City, and we prune the dataset keeping the users from the two cities. The two cities have quite different user distribution which is more uniform in Los Angeles and is more concentrated in New York City (cf. Figure 5). For the locations of envisioned edge clouds, we decide to use the locations of the Starbucks due to the fact that Starbucks shops in a city usually can achieve a decent coverage for users. In addition, the distribution of Starbucks shops actually follows the population density, making them perfect locations for edge cloud deployment in the future.

The dataset pruned for Los Angeles (denoted by Twitter-LA) contains 7553 users in total. We keep the relationship of the users as is in the Twitter social graph. Although the frequency of user interaction is not available with the used dataset, we can actually synthesize it following the real-world distributions revealed by [19]. The CDFs of the synthetically generated frequencies of user interaction (including both "read" and "write") are depicted in Figure 6(a). 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 5(a). We assume that each Starbucks shop will be equipped with an edge cloud that is attached to its AP. The network delay between any two Starbucks APs is measured by their geographical distance. We assume that each user 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 users and we synthesize the frequency of user 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).

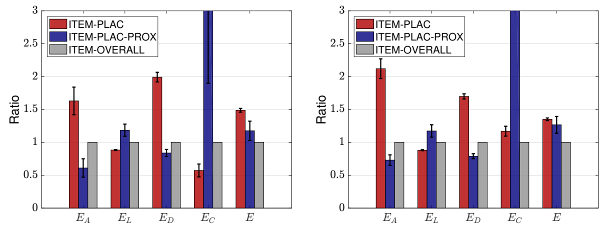
Settings. Our implementation of ITEM is based on a max-flow implementation detailed in [24]. The activation cost for each of the edge clouds is generated randomly following a uniform distribution. For the placement cost, we assume that there are three different price levels and the ratio between the `base` prices of adjacent levels is set to 2. This is to represent the heterogeneity in the cost of using an edge cloud in different areas in the city. The actual placement price for each user placing her SE on an edge cloud is generated following a normal distribution with the average `base` and standard deviation $0.5\times$`base`. The communication frequency f_u of each user is set to be the sum of frequency of access originated from user u. For the colocation cost, we randomly generate the coefficients following a uniform distribution. We normalize the four types of cost to the range $(0, 1]$ and we set the parameters such that all the costs share equal importance. 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). To further understand the impact of ITEM on different types of cost, we also compare three different ITEM-based solutions: ITEM-PLAC (with only the placement cost optimized), ITEM-PLAC-PROX (with both the placement and the proximity cost optimized), and ITEM-OVERALL (with all the costs optimized).

Results. The performance of ITEM, compared with Ran-



(a) Relative cost ratio (b) Convergency

Fig. 7. Performance evaluation on relative cost ratio and converging speed.



(a) Los Angeles (b) New York City

Fig. 8. Total cost breakdown and impact of ITEM on each cost type.

dom and Greedy, is validated in Figure 7(a). 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 as expected and the cost reduction achieved by ITEM can be 150% when compared with Random and 100% when compared with Greedy. This is due to the fact that Random does not consider any of the costs, while Greedy only minimizes part of the proximity cost (from the user to her SE). Figure 7(b) 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 10 iterations in most cases.

Figure 8 illustrates the breakdown of the total cost and studies the impact of ITEM on each cost type. We observe that ITEM-PLAC minimizes the placement cost E_L, while incurring large activation cost E_A and proximity cost E_D as expected. Surprisingly, the colocation cost E_C is not significant. This is mainly because when minimizing the placement cost, SEs are spread out among the edge clouds, which is also favorable to the colocation cost E_C since E_C is characterized with a super-linear function. ITEM-PLAC-PROX tries to minimize both the placement cost and the proximity cost. As we can see that ignoring the colocation cost E_C can be very problematic as E_C is extremely high ($12\times$ more) in the solution by ITEM-PLAC-PROX. Overall, balancing the four types of cost gives the best performance to ITEM. The results confirm our motivation that *all the four types of cost are critically important and should be optimized simultaneously in an edge computing system, which is the major contribution of this paper.*

B. Online Performance

Dataset and settings. For the online case, mobility is of concern. We use the Rome Taxi dataset and synthesize social

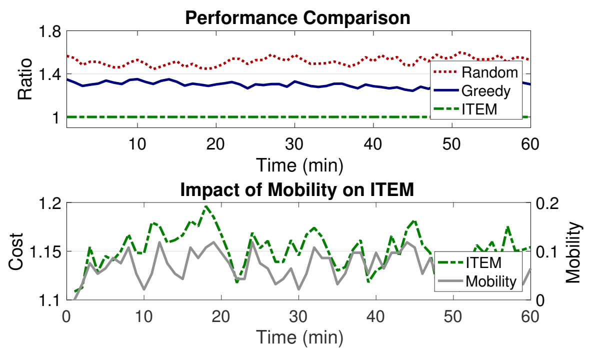


Fig. 9. Performance comparison and impact of mobility in online cases. (The upper plot is normalized by the cost of ITEM, and hence the ratio is always one for ITEM. The actual cost of ITEM fluctuates over time as shown in the lower plot.)

networks with fitted user interactions. We choose a 6-hour period (15h to 20h on date Feb 12, 2014) from the dataset. The number of users varies from hour to hour but is generally around 300 during the considered time period. The users are moving around the city over time and the time granularity is set to one minute. We simulate a social graph on these users following a power-law distribution with exponent 2.5 and we generate the frequency of user interaction using the same approach as for Twitter-LA. We envision an edge computing system with 15 edge clouds that are deployed in the city of Rome and the locations of the edge clouds are chosen from the major metro stations in Rome [11].

We implement a discrete-time simulator where at the beginning of each time slot (e.g., each minute here) we obtain the set of users that have moved and then, we invoke the ITEM algorithm on those users to obtain new placement decisions. We use the above mentioned Rome Taxi dataset and other parameters that are generated following the same settings as in the offline case to feed the simulator. We then compare the results with that of Random as well as of Greedy.

Results. 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 18 as all tests in different hours show similar behaviors. All the values in the plots are averaged among five independent runs. The values in the upper plot are normalized to the values obtained by ITEM. As we can see that ITEM outperforms both Random and Greedy as expected and can achieve an overall cost reduction of around 30% under any circumstances in the simulated scenario. In the lower plot, we explore the impact of user mobility to the performance of ITEM, where we show the costs obtained by ITEM normalized to the smallest value we have seen from the solutions of ITEM and the curve of user mobility ratio measured by the ratio between the number of moved users and of all users. As we can see that, the performance of ITEM is quite stable ($1.1-1.2\times$ the minimum) in general and most of the time a positive correlation between the cost of ITEM and the mobility ratio can be observed, i.e., ITEM performs slightly better with lower mobility. This is due to the fact that the online version of ITEM only reoptimizes the placement for users that have moved. With a small mobility ratio, the system remains mostly unchanged and ITEM tries to control the QoS interruption on the unmoved users.

V. Related Work

Data placement for online social networks. A lot of work has been carried out on optimizing cost or performance via data placement or replication for online social networks [13]–[16]. Jiao et al. propose cosplay, a cost effective data placement policy that can guarantee QoS in online social networks [13]. Later, they further investigate the problem by considering multiple objectives [15]. Yu and Pan propose an associated data placement (ADP) scheme which aims to improve the colocation of associated data and localized data serving [14]. Recently, Zhou et al. explore the problem of joint placement and replication of social network data with the goal of minimizing network traffic [16].

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 [5]–[11]. Jia et al. study the load balancing among multiple edge clouds in [6]. Tong et al. discuss workload placement for delay minimization in a hierarchical edge computing architecture [7]. Wang et al. [8] and Urgaonkar et al. [9] focus on stochastic frameworks for optimizing dynamic workload migration based on Markov Decision Processes (MDPs) and Lyapunov optimization. Recently, Tan et al. study online job dispatching and scheduling in edge clouds [10]. Wang et al. investigate online mobility-oblivious resource allocation for edge computing [11].

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

VI. Conclusion

In this paper we conduct the first formal study of the service entity placement problem for social VR applications in edge computing. 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. The algorithm is flexible and is applicable in both offline and online cases. The performance of the proposed algorithm is confirmed by extensive experiments. As the intersection of VR and edge computing is gaining more and more attention, the solution provided in this paper will serve as a baseline and will foster future exploration in this direction. Several research questions are still open including developing online algorithms for SE placement with edge cloud reconfiguration cost included. We leave them to future work.

Acknowledgement

This work was partially funded by the German Research Foundation (DFG) and the National Nature Science Foundation of China (NSFC) joint project under Grant No. 392046569 (DFG) and No. 61761136014 (NSFC), and the DFG Collaborative Research Center (CRC) 1053 – MAKI. Ting He was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-16-3-0001. Jun Li was funded by the National Science Foundation under Grant No. 1564348. 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] ETSI MEC-IEG, "Mobile edge computing (mec); service scenarios."
[2] Open Edge Computing. http://openedgecomputing.org.
[3] M. Jang, H. Lee, K. Schwan, and K. Bhardwaj, "SOUL: an edge-cloud system for mobile applications in a sensor-rich world," in *SEC*, 2016.
[4] J. Cho, K. Sundaresan, R. Mahindra, J. E. van der Merwe, and S. Rangarajan, "ACACIA: context-aware edge computing for continuous interactive applications over mobile networks," in *CoNEXT*, 2016.
[5] L. Wang, L. Jiao, D. Kliazovich, and P. Bouvry, "Reconciling task assignment and scheduling in mobile edge clouds," in *ICNP*, 2016.
[6] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *INFOCOM*, 2016.
[7] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *INFOCOM*, 2016.
[8] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. S. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Networking*, 2015.
[9] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. S. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *PEVA*, vol. 91, pp. 205–228, 2015.
[10] H. Tan, Z. Han, X.-. Li, and F. C. Lau, "Online job dispatching and scheduling in edge clouds," in *INFOCOM*, 2017.
[11] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *ICDCS*, 2017.
[12] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. M. Lau, "Scaling social media applications into geo-distributed clouds," *ToN*, vol. 23, no. 3, pp. 689–702, 2015.
[13] L. Jiao, J. Li, T. Xu, and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in *ICNP*, 2012.
[14] B. Yu and J. Pan, "Location-aware associated data placement for geo-distributed data-intensive applications," in *INFOCOM*, 2015.
[15] L. Jiao, J. Li, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *INFOCOM*, 2014.
[16] J. Zhou and J. Fan, "JPR: exploring joint partitioning and replication for traffic minimization in online social networks," in *ICDCS*, 2017.
[17] N. Bansal, K. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," in *PODC*, 2011.
[18] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *ToN*, vol. 20, no. 1, pp. 206–219, 2012.
[19] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *EuroSys*, 2009.
[20] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: challenges and approaches," *SIGMETRICS PER*, vol. 37, no. 3, pp. 55–60, 2009.
[21] J. Krarup and P. M. Pruzen, "The simple plant location problem: Survey and synthesis," *Eur. J. Oper. Res.*, vol. 12, pp. 36–81, 1983.
[22] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," in *CVPR*, 2010, pp. 2173–2180.
[23] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts: a review," *IEEE TPAMI*, vol. 29, no. 7, pp. 1274–1279, 2007.
[24] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE TPAMI*, vol. 26, no. 9, pp. 1124–1137, 2004.