Service Placement and Request Scheduling for Data-intensive Applications in Edge Clouds

Vajiheh Farhadi[1], Fidan Mehmeti[2], Ting He[3], Tom La Porta[4], Hana Khamfroush[5], Shiqiang Wang[6], Kevin S Chan[7], Sebastian Stein[8]
[1,2,3,4]Pennsylvania State University, [5]University of Kentucky, [6]IBM, [7]ARL, [8]University of Southampton
[1]vuf8@cse.psu.edu, [2]fzm82@psu.edu, [3]th.tinghe@gmail.com, [4]tlp@cse.psu.edu, [5]khamfroush@cs.uky.edu, [6]wangshiq@us.ibm.com, [7]kevin.s.chan.civ@mail.mil, [8]ss2@ecs.soton.ac.uk

***Abstract*—Mobile edge computing allows wireless users in military networks to exploit the power of cloud computing without the large communication delay. To serve data-intensive applications (e.g., video analytics, augmented reality) from the edge, we need, in addition to CPU cycles and memory for computation, storage resource for storing server data and network bandwidth for receiving user-provided data. Moreover, the data placement needs to be adapted over time to serve time-varying demands, while considering system stability and operation cost. We address this problem by proposing a two-time-scale framework that jointly optimizes service (data & code) placement and request scheduling, under storage, communication, computation, and budget constraints. We fully characterize the complexity of our problem by analyzing the hardness of various cases. By casting our problem as a set function optimization, we develop a polynomial-time algorithm that achieves a constant-factor approximation under certain conditions. Extensive synthetic and trace-driven simulations show that the proposed algorithm achieves 90% of the optimal performance.**

***Index Terms*—Mobile edge computing, service placement, resource allocation, complexity analysis.**

I. Introduction

The emerging technology of *mobile edge computing* [1] enables wireless users to run resource-intensive and delay-sensitive applications from the edge of mobile networks, at small server clusters referred to as *edge clouds* [2], *cloudlets* [3], *fog* [4], *follow me cloud* [5], or *micro clouds* [6]. This is particularly beneficial for military networks, where accessing a remote cloud can be costly and can incur large delays. While the technology is designed to harness the computation power of cloud computing without the large communication delays in accessing remote clouds, realizing the full potential of mobile edge computing requires smart strategies in allocating the limited edge cloud resources to competing requests, which has attracted significant research attention in recent years.

Intuitively, one should strive to serve every user from the nearest edge cloud. While the intuition has been supported by empirical studies [7], maintaining service locality for mobile users raises challenges about how to migrate services [8] and when/where to migrate services [9], [10], [2], [11] to achieve a desirable tradeoff between service performance and migration cost. When some of the edge clouds are heavily loaded, it has been shown that users can benefit from getting served by non-nearest edge clouds in the same metropolitan area network [12], [13], [14]. Meanwhile, there have been standardization initiatives [15], [16], [17] to create a standardized open edge computing environment, such that edge clouds within the same geographical region will form a shared resource pool, which can then be allocated among competing user requests.

The existence of a shared resource pool opens the door to *request scheduling*, i.e., on which edge server, if any, to schedule each user request such that certain objectives (e.g., cost, completion time) can be optimized [18], [19]. Existing works typically assume that serving each request requires a *dedicated* share of resources (e.g., CPU cycles, memory space, network bandwidth), such that the total resource consumption at a server is the sum of resource requirements scheduled to it.

While this assumption holds for applications that do not require significant amounts of data on the server, it fails to capture the demands of *data-intensive applications*. In such applications (e.g., augmented reality, video analytics, distributed machine learning), serving a request not only requires a dedicated share of resources, but also requires a nontrivial amount of data at the server (e.g., object database, trained machine learning models). The storage resource for storing such data critically differs from the other types of resources in that it is *amortized* over all requests against the same copy of data. Note that many data-intensive applications also require a nontrivial amount of user-provided data (e.g., images captured by the user), although the resources for collecting/storing such data are typically dedicated to each request.

Thus, in addition to the traditional resources of CPU cycles and memory, resource allocation algorithms for data-intensive applications must also consider the storage resources for storing server data and the network bandwidth for receiving requests that contain large amounts of user-provided data.

Jointly allocating dedicated resources and amortized resources induces a decomposition of the problem into two subproblems [20]: (i) *service placement*, which decides how to replicate and place each service (including server code and

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. 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 Defence 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.

data) within the storage capacity of each edge cloud, and (ii) *request scheduling*, which decides whether/where to schedule each request within the communication and the computation capacities of edge clouds, as well as other constraints (e.g., maximum delay). The two subproblems are coupled by the fact that the edge cloud scheduled to process a request must have a replica of the requested service. However, the existing solution [20] makes both decisions at the same time, and thus may adjust service placement as frequently as scheduling requests, incurring a high operation cost and even system instability.

In this work, we jointly consider service placement and request scheduling for data-intensive applications. In contrast to [20], we separate the time scales of the two decisions: service placement occurs at a larger scale (*frames*) to prevent system instability, and request scheduling occurs at a smaller scale (*slots*) to support real-time services. We also impose a budget constraint to control the operation cost due to service replication/migration. These changes enable a controllable tradeoff between the cost of reconfiguration and the performance of serving requests, while inducing critical changes in the underlying optimization problem.

A. Summary of Contributions

Our main contributions are as follows:

1) We propose a two-time-scale framework for joint service placement and request scheduling, and formulate the underlying optimization as a mixed integer linear program (MILP) that jointly considers dedicated and amortized resources.

2) By analyzing the complexity in carefully selected special cases, we not only prove that our problem is generally NP-hard, but also characterize all the cases that are polynomial-time solvable and identify the root cause of hardness.

3) By reformulating our problem as a set function optimization, we develop a greedy service placement algorithm based on shadow request scheduling computed by a linear program (LP). By proving that our objective function is monotone submodular under certain conditions and our constraints form a p-independence system, we derive a constant-factor approximation guarantee for the proposed algorithm.

4) We show that both our formulation and our algorithm can be extended to exploit request prediction over multiple frames.

5) We perform extensive performance evaluations via synthetic and trace-driven simulations. The proposed algorithm consistently outperforms baselines, while achieving over 90% of the optimal performance in all the evaluated cases, even when the approximation guarantee does not hold.

II. Problem Formulation

A. System Model

As illustrated in Fig. 1, we consider a wireless edge network consisting of a set N of edge clouds, each accessible via a wireless access point or base station covering a specified area. There is a set L of services, of which a subset can be hosted by each edge cloud at a given point in time. At each time slot t, requests for service $l \in L$ arrive at edge cloud $n \in N$ at a rate of λ^t_{ln}. The average request rate for a frame f of T_f time slots is denoted by λ^f_{ln}, where T_f is chosen to trade off system stability and prediction accuracy for λ^f_{ln}.

Services may migrate/replicate between edge clouds, and from a remote cloud to an edge cloud. Each edge cloud has limited communication, computation, and storage capacities. Furthermore, we impose a budget B on the cost of migrating/replicating services between edge clouds or from the remote cloud to an edge cloud in each frame. We assume that all the edge clouds are connected by back-haul links that can be used for inter-cloud communications. Thus, a request may be served by a non-local edge cloud.

Serving a request for service l requested at edge cloud n and then served by edge cloud m (possibly $m \neq n$) consumes communication resources for transferring input/output between the user and edge cloud n, and computation resource at m. Additionally, edge cloud m must have a replica of service l. As back-haul links usually have much higher bandwidth than access links, we only consider the communication resource consumed at the access link in edge cloud n.

The capacities of different edge clouds may be different. Likewise the size of each service replica and the communication/computation resources required by each request may be different. There may be other constraints (e.g., latency) on whether a given edge cloud m is permitted to serve requests of service l submitted to another edge cloud n, denoted by an indicator a_{lnm} ('0': not permitted; '1': permitted).

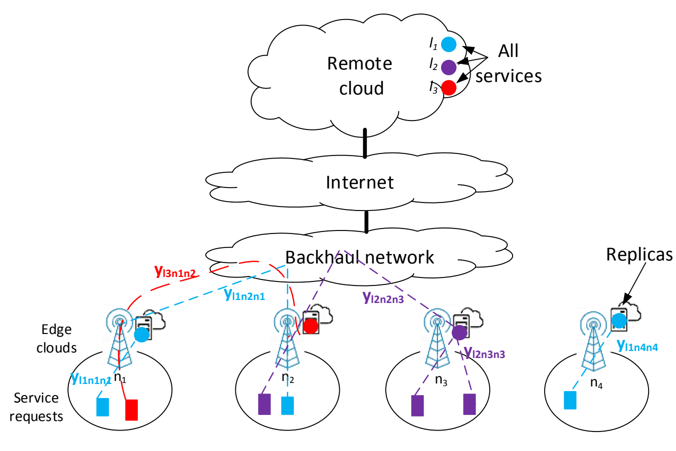

Figure 1. System model.

In order to control the system stability and the costs of placing replicas, we split the time scales of service placement (performed at the beginning of every frame) from request scheduling (performed per slot). This is illustrated in Fig. 2. Finally, main notations used in this paper are described in Table I.

B. Underlying Optimization Problem

Although at different time scales (Fig. 2), service placement and request scheduling are solving the same optimization problem with different decision variables as explained below.

We assume that the services always exist on the remote cloud n_0, i.e., $x^f_{ln_0} \equiv 1$, and deleting a service replica from an edge cloud incurs no cost. We always replicate a service from the nearest location hosting the service. That

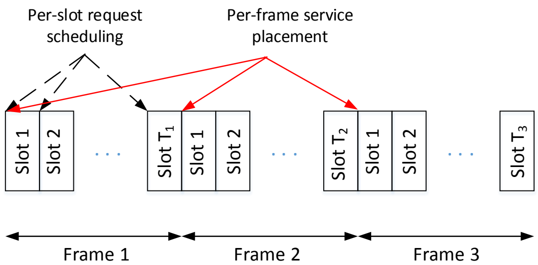

Figure 2. Time scales of service placement and request scheduling.

Table I
TABLE OF NOTATIONS

Notation	meaning
N	set of edge clouds
$N_+ = N \cup \{n_0\}$	set of edge clouds plus the remote cloud n_0
L	set of all possible services
R_n	storage capacity of edge cloud n
W_n	processing capacity of edge cloud n
K_n	communication capacity of edge cloud n
r_l	size per replica of service l
κ_l	size of input/output data per request of service l
ω_l	computation requirement per request of service l
$a_{lnm} \in \{0,1\}$	indicates whether edge cloud m is permitted to serve requests of service l submitted to edge cloud n
$\lambda^t_{ln}, \lambda^f_{ln}$	average arrival rate of requests of service l submitted to edge cloud n in time slot t or frame f (averaged over all the slots in this frame)
$c_{ln'n}$	cost of replicating or migrating service l from cloud n' to edge cloud n, where cloud n' can be either the remote cloud or edge cloud
B	the budget for service placement in one frame
$x^f_{ln} \in \{0,1\}$	placement variable for frame f, 1 if service l is placed on edge cloud n and 0 otherwise
$y^t_{lnm}, y^f_{lnm} \in [0,1]$	scheduling variable representing the probability that a request of service l submitted to edge cloud n is scheduled to another edge cloud m in slot t or frame f

is, the cost of placing service l at edge cloud n in frame f is $c^f_{ln} = \min_{n' \in N_+, x^{f-1}_{ln'}=1} c_{ln'n}$, where $c_{lnn} \equiv 0$ (no replication, no cost).

The underlying optimization problem can be formulated as (1): Objective (1a) maximizes the expected number of requests served per slot. Constraint (1b) guarantees that the scheduling variables are valid probabilities. Constraint (1c) ensures that each edge cloud n does not store more than its storage capacity R_n. Constraint (1d) guarantees that each edge cloud n does not violate its communication capacity K_n within its coverage area. Constraint (1e) ensures that each edge cloud n is not scheduled with more requests than its computation capacity W_n allows. Constraint (1f) states that an edge cloud can only serve a request if it contains the requested service and is a candidate server. Constraint (1g) ensures that the total service placement cost is within the budget. Constraint (1h) specifies valid ranges of the decision variables.

$$\max \sum_{l\in L}\sum_{n\in N} \lambda_{ln} \sum_{m\in N} y_{lnm} \quad (1a)$$

$$\text{s.t.} \sum_{m\in N} y_{lnm} \le 1, \quad \forall l \in L, n \in N, \quad (1b)$$

$$\sum_{l\in L} x_{lm} r_l \le R_m, \quad \forall m \in N, \quad (1c)$$

$$\sum_{l\in L} \lambda_{ln}\kappa_l \sum_{m\in N} y_{lnm} \le K_n, \quad \forall n \in N, \quad (1d)$$

$$\sum_{l\in L} \omega_l \sum_{n\in N} \lambda_{ln} y_{lnm} \le W_m, \quad \forall m \in N, \quad (1e)$$

$$y_{lnm} \le a_{lnm} x_{lm}, \quad \forall l \in L, n \in N, m \in N, \quad (1f)$$

$$\sum_{l\in L}\sum_{n\in N} x_{ln} c_{ln} \le B, \quad (1g)$$

$$x_{ln} \in \{0,1\}, y_{lnm} \ge 0, \quad \forall l \in L, n \in N, m \in N. \quad (1h)$$

At the beginning of each frame f, we solve (1) with the predicted demands $\lambda_{ln} = \lambda^f_{ln}$ and the placement costs $c_{ln} = c^f_{ln}$ for the service placement x^f_{ln} and the corresponding request scheduling y^f_{lnm}. Then at the beginning of each slot t, we solve (1) with the current demands $\lambda_{ln} = \lambda^t_{ln}$ and the previously determined service placement $x_{ln} = x^f_{ln}$ for the request scheduling y^t_{lnm} used in this slot. Note that although the scheduling variable y^f_{lnm} computed from the predicted demands is not used for scheduling, it is needed to evaluate the objective (1a) under a given service placement. For this reason, we refer to y^f_{lnm} as the *shadow scheduling variable*.

Discussion: While our optimization formulation shares similarities with [20], there are several critical changes. First, while [20] assumes full knowledge of the requests, we only assume knowledge of the expected request rates. Accordingly, our objective becomes the expected rate of served requests, and our scheduling decision becomes probabilistic. Moreover, while [20] allows the service placement to change completely every time, we limit it to incremental adjustments by imposing a budget constraint. Probabilistic scheduling relaxes the integer constraints on scheduling variables, thus invalidating previous hardness results. Meanwhile, the added constraint introduces a potential cause of hardness (verified in Theorem 1).

III. COMPLEXITY ANALYSIS

In the optimization problem (1) there are four types of resource constraints: the R-constraint (1c), the K-constraint (1d), the W-constraint (1e), and the B-constraint (1g).

A. Having B-constraint only

Consider the special case where the edge clouds and the services are homogeneous (although having B-constraint only gives the same formulation for homogeneous and heterogeneous scenarios), and R, W and K are large enough that they are unconstrained, i.e., $R \ge |L|$ (i.e., every edge cloud can store all the services), $W \ge \sum_{n\in N}\sum_{l\in L} \lambda_{ln}$ and $K \ge \max_{n\in N} \sum_{l\in L} \lambda_{ln}$. Then, the MILP in (1) changes to:

$$\max \sum_{l\in L}\sum_{n\in N} \lambda_{ln} \sum_{m\in N} y_{lnm} \quad (2a)$$

$$\text{s.t. } (1b), (1f), (1g), \quad (2b)$$

$$x_{ln} \in \{0,1\}, y_{lnm} \in [0,1], \quad \forall l \in L, n \in N, m \in N. \quad (2c)$$

Theorem 1. *The B-constraint alone makes the problem NP-hard.*

Proof. We prove the NP-hardness of (2) by a reduction from the 0-1 knapsack problem: given a set of k items, each with value v_i and weight w_i $(i = 1, ..., k)$, select a subset $\mathcal{S}'$ such that $\sum_{i\in\mathcal{S}'} v_i$ is maximized while $\sum_{i\in\mathcal{S}'} w_i \le \Omega$, for a given size Ω of the knapsack.

Construction: For each item i, construct a service l_i with total demands $\sum_{n\in N} \lambda_{l_i n} = v_i$ and the placement cost $c_{l_i n} = w_i, \forall n \in N$. Let $B = \Omega$ and $a_{lmn} \equiv 1$.

Claim: The optimal service placement of (2) gives the optimal solution to a knapsack problem.

Proof of the claim: The optimal service placement places at most one replica among all the edge clouds. Therefore, the scheduling decision is to simply schedule all the requests of service l_i to edge cloud n, if $\exists n \in N$ with $x_{l_i n} = 1$; or, not schedule any of these requests if $x_{l_i n} = 0, \forall n \in N$. Let $\mathcal{S}'$ be the set of indices of all the placed services under the optimal solution to (2). Then, the expected number of served requests equals $\sum_{i\in\mathcal{S}'} v_i$, and $\sum_{i\in\mathcal{S}'} w_i \le B = \Omega$. Selecting all the items corresponding to the services placed by the optimal solution of (2) provides the optimal solution to the knapsack problem. □

Remark: Proving NP-hardness for the special case shows that the problem is NP-hard in the general case as well.

B. Having R-constraint only

Here we consider the special case in which the edge clouds and the services are homogeneous, and W, K and B are large enough to be unconstrained, i.e., $W \ge \sum_{n\in N} \sum_{l\in L} \lambda_{ln}$, $K \ge \max_{n\in N} \sum_{l\in L} \lambda_{ln}$, and $B \ge \sum_{l\in L}\sum_{n\in N} c_{ln}$. In this case, the MILP in (1) becomes:

$$\max \sum_{l\in L}\sum_{n\in N} \lambda_{ln} \sum_{m\in N} y_{lnm} \tag{3a}$$

$$\text{s.t. } (1b), (1f), (2c), \tag{3b}$$

$$\sum_{l\in L} x_{ln} \le R, \qquad \forall n \in N. \tag{3c}$$

Theorem 2. *The R-constraint alone makes the problem NP-hard.*

Proof. We prove the hardness by showing that the optimization (3) can be reduced to the 2-Disjoint Set Cover ($2DSC$) problem, which is proved to be NP-complete [21]. Given a bipartite graph $\mathcal{G} = (\mathcal{A}, \mathcal{B}, \mathcal{E})$, with edges $\mathcal{E}$ between two disjoint vertex sets $\mathcal{A}$ and $\mathcal{B}$, 2DSC determines whether there exist two disjoint sets $\mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$, such that $|\mathcal{B}_1| + |\mathcal{B}_2| = |\mathcal{B}|$ and $\mathcal{A} = \cup_{b\in\mathcal{B}_1}\mathcal{N}(b) = \cup_{b\in\mathcal{B}_2}\mathcal{N}(b)$, where $\mathcal{N}(b)$ $(\forall b \in \mathcal{B})$ is the set of neighbors of node b.

Construction: Denote $\mathcal{A}$ by $\{a_1, ..., a_I\}$ and $\mathcal{B}$ by $\{b_1, ..., b_J\}$. WLOG, assume $I \le J$. Construct J edge clouds $N = \{n_1, ..., n_J\}$, each with $R = 1$. Construct two services $L = \{l_1, l_2\}$, each with a unit of demand in the first I edge clouds, i.e., $\lambda_{ln_i} = 1, \forall i \in \{i, ..., I\}, l \in \{l_1, l_2\}$. Note that $\lambda_{ln_i} = 0, \forall i > I$. For each $i \in \{1, ..., I\}$ and $j \in \{1, ..., J\}$, we allow edge cloud n_j to serve requests of either service in edge cloud n_i, if and only if $(a_i, b_j) \in \mathcal{E}$, i.e., $a_{l_k n_i n_j} = 1,\ k = \{1, 2\}$, if $(a_i, b_j) \in \mathcal{E}$, otherwise $a_{l_k n_i n_j}$ is zero.

Claim: $2DSC$ is feasible if and only if the optimal value of (3) for the above instance is $2\mathcal{I}$.

Proof of the claim: If $2DSC$ is feasible, then storing l_1 at edge clouds corresponding to $\mathcal{B}_1$ and l_2 at the remaining edge clouds will serve all the requests. If there is a service placement that serves all the requests, then $\mathcal{B}_1 = \{b_i \in \mathcal{B} : n_i \text{ stores } l_1\}$, and $\mathcal{B}_2 = \mathcal{B}\backslash\mathcal{B}_2$ is a feasible solution to $2DSC$. □

C. Removing R- and B-constraints

If R_n $(\forall n \in N)$ and B are both large enough, i.e., $\min_{n\in N} R_n \ge |L|$ (every edge cloud can store all the services) and $B \ge \sum_{l\in L}\sum_{n\in N} c_{ln}$, the optimal solution to x_{ln} is trivially $x_{ln} \equiv 1$ $(\forall l \in L$ and $n \in N)$. Under this service placement, constraints (1c,1g) in (1) disappear, and constraint (1f) changes to $y_{lnm} \le a_{lnm}$ $(\forall l \in L, n \in N, m \in N)$.

Lemma 1. *Removing R- and B-constraints makes the problem polynomial-time solvable.*

Proof. Removing these constraints reduces the original problem (1) into an LP, which is polynomial-time solvable. □

D. Summary of all cases

Together, Theorems 1, 2 and Lemma 1 cover all the cases. By Theorem 1, the solvable instances must be cases without the B-constraint. By Theorem 2, the solvable instances must also be cases without the R-constraint. On the other hand, Lemma 1 shows that all the cases without either of B- or R-constraint are polynomial-time solvable. Therefore, the colored region in Fig. 3 captures all the solvable cases of (1).

IV. ALGORITHMS

We now develop efficient algorithms for the service placement and the request scheduling sub-problems separately.

A. Optimal Algorithm for Request Scheduling

Recall that request scheduling is performed at a smaller time scale of slots, under the service placement selected at the beginning of each frame (see Section II). Given the request rates observed at the beginning of each slot, we can solve the sub-problem of (1) regarding $\mathbf{y}$, which is an LP (see (4)), and perform probabilistic scheduling, where a request for service l submitted to edge cloud n will be scheduled to edge cloud m for each $m \in N$ with probability y_{lnm}. All the scheduling decisions in a frame are based on the same service placement, while the decisions in different slots can differ due to variations in request rates within the frame.

B. Approximation Algorithm for Service Placement

Due to the NP-hardness of finding the optimal service placement, as shown in Section III, we seek efficient sub-optimal service placement algorithms with approximation guarantees.

We start by reformulating our problem as a set optimization problem. Let $S \subseteq L \times N$ denote the set of selected single-service placements, where $(l, n) \in S$ means to place a replica

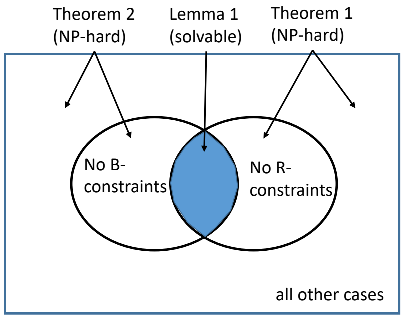

Figure 3. Complexity of (1) and its cases.

of service l at edge cloud n. Let $\Omega(S)$ denote the optimal objective value of (1) for a fixed $\mathbf{x}$ given by $x_{ln} = 1$ if and only if $(l,n) \in S$. This can be calculated by solving the following (shadow) request scheduling problem, where $\mathbb{1}_{l,m}$ is the indicator function:

$$\max \sum_{l\in L}\sum_{n\in N} \lambda_{ln} \sum_{m\in N} y_{lnm} \tag{4a}$$
$$\text{s.t. } (1b), (1d), (1e), \tag{4b}$$
$$y_{lnm} \le a_{lnm}\mathbb{1}_{(l,m)\in S}, \quad \forall l \in L, n \in N, m \in N, \tag{4c}$$
$$y_{lnm} \in [0,1], \quad \forall l \in L, n \in N, m \in N. \tag{4d}$$

After that, we can rewrite the service placement problem as:

$$\max \Omega(S) \tag{5a}$$
$$\text{s.t. } \sum_{l:(l,n)\in S} r_l \le R_n, \quad \forall n \in N, \tag{5b}$$
$$\sum_{(l,n)\in S} c_{ln} \le B, \tag{5c}$$
$$S \subseteq L \times N, \tag{5d}$$

where $S_n \triangleq L \times \{n\}$ is the set of all possible single-service placements at edge cloud n.

First, we prove that, under certain conditions, the objective function of (5) has a desirable property.

Definition 1 ([22]). *A set function $f : 2^{\mathbf{x}} \to \mathcal{R}$ is monotone increasing if $\forall\ S_1 \subseteq S_2 \subseteq \mathbf{x}$, $f(S_1) \le f(S_2)$. Moreover, the function $f(.)$ is sub-modular if $\forall S_1 \subseteq S_2 \subseteq \mathbf{x}$ and $e \in \mathbf{x}\backslash S_2$, $f(\{e\} \cup S_1) - f(S_1) \ge f(\{e\} \cup S_2) - f(S_2)$.*

Lemma 2. *The objective function $\Omega(S)$ in (5a) is a monotone sub-modular function for all feasible S if $\kappa_l \equiv \kappa$ ($\forall l \in L$), and*

1) *$\lfloor R_n/r_l \rfloor \le 1$ for all $n \in N$ and $l \in L$, or*
2) *$W_m \ge \sum_{l\in L} \omega_l \sum_{n\in N} \lambda_{ln}$ for all $m \in N$.*

Proof. It is easy to see that $\Omega(S)$ is monotone, as expanding S will relax the constraint (4c), hence enlarge the solution space for (4) and increase its optimal objective value.

To show that $\Omega(S)$ is sub-modular, we need to show that for any sets $S_1, S_2 \subseteq L \times N$ and any $(l_1, n_1) \in (L \times N) \setminus S_2$, such that $S_1 \subseteq S_2$ and $S_2 \cup \{(l_1, n_1)\}$ is feasible, the following relationship holds

$$\Omega(S_1 \cup \{(l_1, n_1)\}) - \Omega(S_1) \ge \Omega(S_2 \cup \{(l_1, n_1)\}) - \Omega(S_2). \tag{6}$$

Suppose that $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(2)}$ are the optimal scheduling solutions according to (4) under service placements S_1 and S_2, respectively. Moreover, suppose that $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(3)}$ are the optimal scheduling solutions under service placements $S_1 \cup \{(l_1, n_1)\}$ and $S_2 \cup \{(l_1, n_1)\}$, respectively, that minimize the request rate scheduled to the replica (l_1, n_1), i.e., minimizing $\sum_{n\in N} \lambda_{l_1 n} y_{l_1 n n_1}$. We can then decompose the objective function as:

$$\Omega(S_1) = \sum_{(l,m)\in S_1} \sum_{n\in N} \lambda_{ln} y^{(0)}_{lnm}, \tag{7}$$
$$\Omega(S_1 \cup \{(l_1, n_1)\}) = \sum_{(l,m)\in S_1} \sum_{n\in N} \lambda_{ln} y^{(1)}_{lnm} + \sum_{n\in N} \lambda_{l_1 n} y^{(1)}_{l_1 n n_1}, \tag{8}$$
$$\Omega(S_2) = \sum_{(l,m)\in S_2} \sum_{n\in N} \lambda_{ln} y^{(2)}_{lnm}, \tag{9}$$
$$\Omega(S_2 \cup \{(l_1, n_1)\}) = \sum_{(l,m)\in S_2} \sum_{n\in N} \lambda_{ln} y^{(3)}_{lnm} + \sum_{n\in N} \lambda_{l_1 n} y^{(3)}_{l_1 n n_1}. \tag{10}$$

Due to this decomposition, we have

$$\text{LHS of (6)} = \sum_{(l,m)\in S_1} \sum_{n\in N} \lambda_{ln} (y^{(1)}_{lnm} - y^{(0)}_{lnm}) + \sum_{n\in N} \lambda_{l_1 n} y^{(1)}_{l_1 n n_1}, \tag{11}$$
$$\text{RHS of (6)} = \sum_{(l,m)\in S_2} \sum_{n\in N} \lambda_{ln} (y^{(3)}_{lnm} - y^{(2)}_{lnm}) + \sum_{n\in N} \lambda_{l_1 n} y^{(3)}_{l_1 n n_1}. \tag{12}$$

The first term in (11) is the difference in the request rate served by replicas in S_1 after/before placing the replica (l_1, n_1). Under condition (1) or (2) in the lemma, there is no contention of computation resources between replicas, and hence replicas in S_1 can still process requests scheduled to them under $\mathbf{y}^{(0)}$. Meanwhile, as the communication demands κ_l are the same for all types of requests, dropping requests originally scheduled to S_1 to admit requests to be scheduled to (l_1, n_1) will not improve the objective value of (4). Thus, the first term in (11) is zero. Similarly, the first term in (12) is also zero. The second term in (11,12) is the minimum request rate served by the replica (l_1, n_1) under an optimal scheduling, in the presence of replicas S_1 and S_2, respectively. Again, as there is no computation resource contention between replicas, requests that used to be served by replicas in S_1 under service placement $S_1 \cup \{(l_1, n_1)\}$ can still be served there after adding replicas in $S_2 \setminus S_1$, but these added replicas may offload some requests that used to be served by the replica (l_1, n_1). Therefore, $\sum_{n\in N} \lambda_{l_1 n} y^{(1)}_{l_1 n n_1} \ge \sum_{n\in N} \lambda_{l_1 n} y^{(3)}_{l_1 n n_1}$. This proves (6) and hence the sub-modularity of $\Omega(S)$. □

The constraints of (5) also have a desirable property.

Definition 2 ([23]). *Let X be a universe of elements. Consider a collection $\mathcal{I} \subseteq 2^X$ of subsets of X. $(X, \mathcal{I})$ is called an independence system if: (a) $\emptyset \in \mathcal{I}$, and (b) if $Z \in \mathcal{I}$ and $Y \subseteq Z$, then $Y \in \mathcal{I}$ as well. The subsets in $\mathcal{I}$ are called independent; for any set S of elements, an inclusion-wise maximal subset T of S that is in $\mathcal{I}$ is called a basis of S.*

Definition 3 ([23]). *Given an independence system $(X, \mathcal{I})$ and a subset $S \subseteq X$, the rank $r(S)$ is defined as the cardinality of the largest basis of S, and the lower rank $\rho(S)$ is the cardinality of the smallest basis of S. The independence system is called a p-independence system (or a p-system) if $\max_{S\subseteq X} \frac{r(S)}{\rho(S)} \le p$.*

Algorithm 1: Greedy Service Placement based on Shadow Scheduling (GSP-SS)

Input: Input parameters of (1)
Output: Service placement $\mathbf{x}$ and request scheduling $\mathbf{y}$

$S \leftarrow \emptyset$;
$\omega^* \leftarrow 0$;
while $\exists\ (l,n) \in (L \times N) \setminus S$ such that $S \cup (l,n)$ satisfies (5b)-(5d) **do**
  $(l^*, n^*) \leftarrow \arg\max_{(l,n):\ S\cup\{(l,n)\} \text{ satisfies (5b)-(5d)}} \Omega(S \cup \{(l,n)\})$;
  $S \leftarrow S \cup \{(l^*, n^*)\}$;
Convert S to its vector representation $\mathbf{x}$;
Compute $\mathbf{y}$ by solving LP for input $\mathbf{x}$;

Lemma 3. *The constraints (5b)-(5d) form a p-independence system for* $p= \lceil \frac{\max\ c_{ln}}{\min_{c_{ln}>0}\ c_{ln}} \rceil + \lceil \frac{\max\ r_l}{\min_{l:r_l>0}\ r_l} \rceil$.

Proof. By definition 1, $(L \times N, \mathcal{I})$, where $\mathcal{I} \subseteq 2^{L\times N}$ is a set of all feasible solutions to (5) is an independent system, as $S = \emptyset$ is a feasible service placement, and the subset of any feasible service placement remains feasible. Consider any $S \subseteq L \times N$ and any two maximal feasible service placements $S_1,\ S_2 \subseteq S_1$. To add a pair $(l,n) \in S_2 \setminus S_1$ to S_1, we need to take out a set S' of pairs form S_1, such that $(S1 \setminus S') \cup \{(l,n)\}$ remains a feasible service placement. The set S' contains at most $\lceil \frac{\max\ r_l}{\min_{l:r_l>0}\ r_l} \rceil$ pairs from $\{l\} \times N$ corresponding to removing service replicas from edge cloud n to satisfy (5b), and at most $\lceil \frac{\max\ c_{ln}}{\min_{c_{ln}>0}\ c_{ln}} \rceil$ other pairs that correspond to removing service replicas with non-zero placement costs to satisfy (5c). Note that in the worst case all the existing service replicas under S_1 at edge cloud n have zero placement cost, and hence we need to remove replicas at other edge clouds to satisfy the budget constraint (5c). Repeating this swap to each pair in $S_2 \setminus S_1$ shows that we reduce the number of placed service replicas by at most p-fold in modifying S_1 into S_2. Since the above holds for any $S \subset L \times N$ and any maximal independent subsets of S, the constraints (5b)-(5d) form a p-independent system. □

Combining Lemmas 2 and 3 gives the following result.

Theorem 3. *Under the conditions in Lemma 2,* Greedy Service Placement based on Shadow Scheduling *(Algorithm 1) yields a $1/(1+p)$-approximation for (1), where* $p= \lceil \frac{\max\ c_{ln}}{\min_{c_{ln}>0}\ c_{ln}} \rceil + \lceil \frac{\max\ r_l}{\min_{l:r_l>0}\ r_l} \rceil$.

Proof. From [22], for maximizing a monotone sub-modular function subject to a p-system constraint, the greedy algorithm has an approximation ratio of $1/(p+1)$. □

C. Complexity

There are $O(|N| \times \frac{R_{max}}{r_{min}})$ iterations in Algorithm 1, where $R_{max} = \max_{n\in N}\ R_n$ and $r_{min} = \min_{l\in L: r_l>0}\ r_l$. For each iteration, the algorithm considers $O(|L| \times |N|)$ single service placements, and for each single service placement, we need to evaluate the objective function by solving an $O(|N|^{11} \times |L|^{5.5})$ complexity request scheduling sub-problem [24]. Therefore, the overall complexity of Algorithm 1 is $O(|N|^{13} \times |L|^{6.5} \times \frac{R_{max}}{r_{min}}) = O(|N|^{13} \times |L|^{6.5})$.

V. Extension to Multi-frame Optimization

So far we have considered only one frame, with the assumption that our solution will be applied on a frame-by-frame basis. However, for recurrent workloads, it is possible to predict the request rates for a larger time window (e.g., 24 hours) that contains multiple frames, each being a time interval with constant request rates. In this case, the frame-by-frame optimization framework can incur sub-optimality.In [25], we address the proposed multi-frame problem and propose the optimization algorithm to solve it.

VI. Performance Evaluation

We have evaluated the performance of the proposed algorithms using both synthetic and trace-driven simulations.

A. Benchmarks

To assess the performance of our algorithm (GSP-SS), we use the following benchmarks:

1) *the optimal solution* of (1) using an ILP solver;
2) *LP-relaxation with rounding*, which first solves the LP relaxation of (1), and then rounds the placement variables to $\{0,1\}$, subject to R- and B-constraints;
3) *the top-K solution*, which sequentially considers each edge cloud $m \in N$, computes the total demand for each service l that can be scheduled to m, defined as $\Lambda_{lm} = \sum_{n\in N} \lambda_{ln} a_{lnm}$, and then places services at m in descending order of Λ_{lm} until reaching R_m or exhausting the budget.

The performance of every solution is evaluated by the optimal objective value of (4) for the given service placement.

B. Simulation Setup

For synthetic simulations, unless stated otherwise, we set $|N| = 6$ and $|L| = 100$. The values for R_n, K_n and W_n ($\forall n \in N$) are drawn uniformly from the intervals $[24, 36]$, $[16, 24]$ and $[32, 48]$, respectively.[1] Assuming that the edge clouds are associated with hexagon cells arranged into two rows, we set the costs of replicating a service from an edge cloud k hops away or the remote cloud to $0.2k$ and 2, respectively, and the budget B to $0.2 \cdot |N| \cdot |L|$. We set a_{lnm} such that each request can only be served by edge clouds within 2 hops of the edge cloud it is submitted to. The arrival rate of request l is obtained as $\lambda_{ln} = \lambda_n p_{ln}$, where λ_n (total request rate in edge cloud n) is drawn randomly from the interval $[3, 5]$. For p_{ln} (popularity of service l in edge cloud n), we draw a random subset of services L_n, and set $p_{ln} \propto i_l^{-\alpha}$ for each $l \in L_n$, where i_l is the rank of l in L_n, and $\alpha = 0.5$ is the skewness parameter of Zipf's distribution. We initialize the system by randomly placing $|L|/8$ services. For each service l, κ_l, ω_l and r_l are drawn uniformly from $[0.5, 1]$. All results are averaged over 50 Monte Carlo runs.

[1] The values of κ_l and K_n are in KBps, r_l and R_n in TB, and ω_l as well as W_n in Mflops/s.

For the trace-driven simulation, we extract user and edge cloud locations from real mobility traces and cell tower locations. We use the *taxicab* traces from [26], by extracting the traces of 36 users over a 520-minute period with location updates every 10 minutes. Every frame consists of 4 time slots, with each slot lasting for 10 minutes. We assign users into Voronoi cells based on cell tower locations obtained from `http://www.antennasearch.com`, from which we select a subset of 6 cell towers that are at least 9.5 km apart to represent the locations of edge clouds. User requests are generated from a wireless trace from [27], containing transmission timestamps generated by 5 different applications from 36 wireless devices. We associate each device with a user in the taxicab trace, and duplicate each trace 5 times to obtain $|L| = 25$ services. As each timestamp in the original trace represents a single packet, we stretch the time axis by 60 (by treating the time unit as 'minute' instead of 'second') to simulate the arrival process of service requests. The obtained request rates range from 4,330 to 486,841, with a mean of 45,254 (requests/slot). For each edge cloud, we randomly choose a storage capacity R_n of 3-6 TB, a communication capacity of 16-48 Mbps (i.e., $K_n \in [1.2, 3.6]$ GB/slot), and a computation capacity of 50-100 Gflops/sec (i.e., $W_n \in [30, 60]$ Tflops/slot), where a slot $= 10$ min. The other parameters are as before.

C. Results

Synthetic simulation: We compare the performance of different algorithms when varying different input parameters, one at a time, via synthetic simulations.

Fig. 4 illustrates the effect of increasing the computation capacity on the percentage of served requests. As expected, an increase in the computation capacity W leads to a higher percentage of served requests. When comparing the performance of different algorithms for the same computation capacity, we can observe from Fig. 4 that GSP-SS considerably outperforms LP-relaxation with rounding and top-K. Furthermore, it is very close to the optimal solution. We have verified that GSP-SS achieves over 90% of the optimal performance, i.e., the ratio of served requests when using GSP-SS over the optimal solution is greater than 0.9 on the average. Similar observations have been made in the other simulations as well.

We note that our simulation setup does not satisfy the condition in Theorem 3, e.g., κ_l's are different, and thus the theoretical approximation guarantee does not apply. Nevertheless, we have observed empirically that GSP-SS always yields near-optimal performance.

In Figs. 5-7, we vary the other resource parameters, including the budget, the storage capacity, and the communication capacity. Conclusions similar to Fig. 4 follow. Namely, increasing these parameters improves the performance. This trend is more obvious in Fig. 5 and Fig. 6, as these resources (budget and storage capacity) directly affect the set of feasible service placements.

We further vary parameters of request generation. Fig. 8 shows that as we increase the average request rate, the percentage of served requests decreases notably due to the contention of resources. Fig. 9 shows that as we increase the skewness of service popularities by increasing α, the optimal solution and GSP-SS remain the same, while the baselines (LP relaxation with rounding and top-K) improve slightly.

Trace-driven simulation: We cross-validate our observations in a more realistic scenario driven by real traces, as described in Section VI-B. Fig. 10 shows the performance of each algorithm over time. 'Predicted' values are the predicted percentage of served requests when solving (1) at the beginning of each frame. 'Actual' values are the expected percentage of requests served in each slot, computed by (4) for the requests arrived in that slot and the service placement of the corresponding frame. As observed from Fig. 10, GSP-SS closely approximates the optimal not only in the predicted performance but also in the actual performance, while always outperforming the baselines.

Finally, we evaluate the extended GSS-SS for multi-frame optimization. Fig. 11 shows the performance based on request prediction over an $|F|$-frame sliding window. We skip the other algorithms, as top-K performs the same as in Fig. 10, the optimal solution is hard to compute due to nonlinearity of multi-frame optimization problem, and LP relaxation does not apply. As in Fig. 10, 'predicted' values are based on the request rates predicted at the beginning of each window, and 'actual' values are based on the actual request rates in each slot. We see that prediction over a larger window improves the performance of GSS-SS (in terms of actual values), and 2-frame prediction suffices.

VII. Conclusions and Future Work

We proposed a two-time-scale solution for joint service placement and request scheduling in edge clouds under communication, computation, and storage constraints. We not only proved the NP-hardness of the problem in the general case, but also characterized its complexity in all special cases. By combining the greedy heuristic with shadow request scheduling, we developed a polynomial-time service placement algorithm, which was proved to give a constant approximation ratio under certain conditions. Extensive simulations showed that the proposed algorithm achieves near-optimal performance.

In future work, we will consider settings where requests have different levels of importance (or weights). This is particularly important in military settings, where some analytics tasks may be more important to the success of a mission than others or where high-ranked users need to be given priority. This raises new challenges, because probabilistic information about weight distributions needs to be considered during service placement. To deal with this, we will investigate the application of reinforcement learning algorithms to make adaptive decisions even when limited information about these distributions is available.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, March 2017.

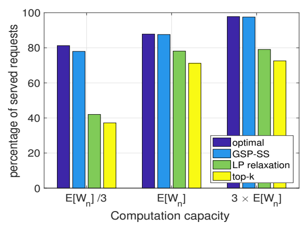

Figure 4. Performance comparison under varying W.

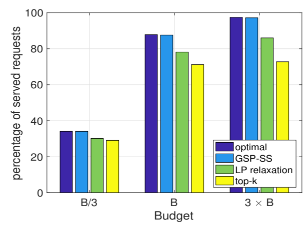

Figure 5. Performance comparison under varying B.

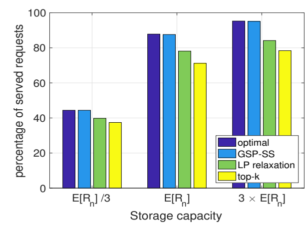

Figure 6. Performance comparison under varying R.

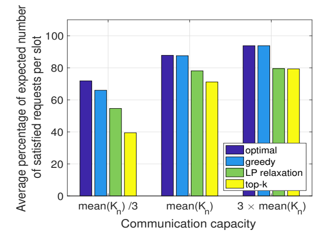

Figure 7. Performance comparison under varying K.

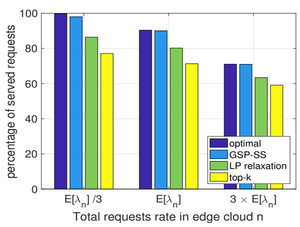

Figure 8. Performance comparison under varying λ.

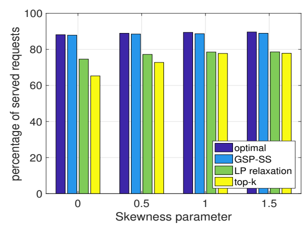

Figure 9. Performance comparison under varying α.

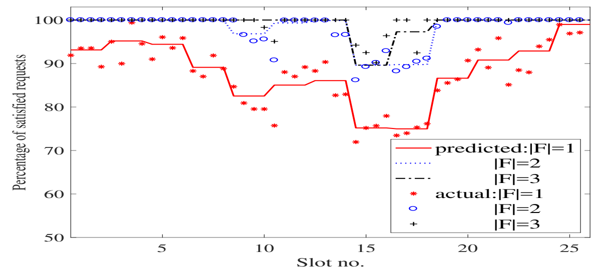

Figure 11. Performance of extended GSS-SS for multi-frame optimization in trace-driven simulation.

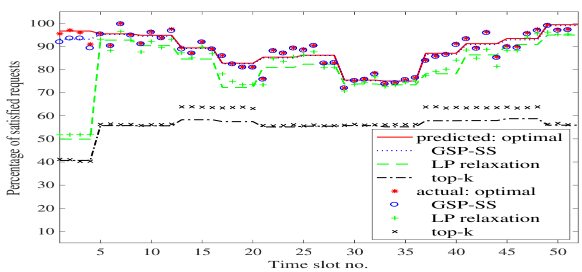

Figure 10. Performance comparison in trace-driven simulation.

[2] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *IFIP Networking*, May 2015.

[3] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49, October 2013.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *MCC*, 2012.

[5] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, September 2013.

[6] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, April 2017.

[7] K. Ha, Y. Abe, Z. Chen, W. He, B. Amos, P. Pillai, and M. Satyanarayanan, "Adaptive VM handoff across cloudlets," Technical Report CMU-CS-15-113, June 2015. [Online]. Available: https://www.cs.cmu.edu/ satya/docdir/CMU-CS-15-113.pdf

[8] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Pillai, and M. Satyanarayanan, "You can teach elephants to dance: Agile VM handoff for edge computing," in *ACM/IEEE Symposium on Edge Computing (SEC)*, October 2017.

[9] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for Follow Me cloud," in *IEEE ICC*, 2014.

[10] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *IEEE MILCOM*, October 2014.

[11] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[12] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[13] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, Oct 2016.

[14] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Trans. on Netw.*, vol. 25, no. 3, 2017.

[15] "Open Edge Computing." [Online]. Available: http://openedgecomputing.org/

[16] "OpenFog Consortium." [Online]. Available: https://www.openfogconsortium.org/

[17] "ETSI ISG on Multi-access Edge Computing (MEC)." [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

[18] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *IEEE ICDCS*, 2017.

[19] H. Tan, Z. Han, X.-Y. Li, and F. Lau, "Online job dispatching and scheduling in edge-clouds," in *IEEE INFOCOM*, May 2017.

[20] T. He, H. Khamfroush, S. Wang, T. L. Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *IEEE ICDCS*, July 2018.

[21] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, p. 333–340, 2005.

[22] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions – II," *Math. Prog. Study*, vol. 8, pp. 73–87, 1978.

[23] A. Gupta, A. Roth, G. Schoenebeck, and K.Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms," *Lecture Notes in Computer Science (LNCS)*, vol. 6484, no. 12, 2010.

[24] G. Strang, "Karmarkar's algorithm and its place in applied mathematics," *The Mathematical Intelligencer*, 1987.

[25] V. Farhadi, F. Mehmeti, T. La Porta, T. He, H. Khamfroush, S. Wang, and K. S Chan, "Service placement and request scheduling for data-intensive applications in edge clouds," 04 2019.

[26] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," February 2009. [Online]. Available: http://crawdad.org/epfl/mobility/20090224

[27] A. S. Uluagac, "CRAWDAD dataset gatech/fingerprinting (v.2014-06-09)," Downloaded from https://crawdad.org/gatech/fingerprinting/20140609/isolatedtestbed, Jun. 2014, traceset: isolatedtestbed.