Joint Service Placement and Request Routing in Mobile Edge Networks with Storage, Computation, and Communication Constraints

Konstantinos Poularakis[1], Ian Taylor[2], and Leandros Tassiulas[1]

[1]Department of Electrical Engineering and Institute for Network Science, Yale University, USA

[2]School of Computer Science and Informatics, Cardiff University, UK

***Abstract*—Modern tactical networks need to provide computation-intensive and latency-sensitive services, often involving different coalition teams, that cannot be supported solely by existing centralized cloud systems. Mobile Edge Computing (MEC) is expected to be an effective solution to meet the demand for low-latency services by enabling the execution of computing tasks in edge servers, close to the end-users. While a number of recent studies have addressed the problem of determining the execution of service tasks and the routing of user requests to corresponding edge servers, the focus has primarily been on the efficient utilization of computing resources, neglecting the fact that non-trivial amounts of data need to be pre-stored to enable service execution, and that many emerging services exhibit asymmetric bandwidth requirements. To fill this gap, we study the joint optimization of service placement and request routing in MEC networks with multidimensional constraints. We show that this problem generalizes several well-known placement and routing problems and propose an algorithm that achieves close-to-optimal performance using a randomized rounding technique. Evaluation results demonstrate that our approach can effectively utilize available storage-computation-communication resources to maximize the number of requests served by low-latency edge cloud servers.**

I. Introduction

A. Motivation

Emerging distributed cloud architectures, such as *Fog* and *Mobile Edge Computing (MEC)*, push substantial amounts of computing functionality to the edge of the network, in proximity to the end-users, thereby allowing to bypass fundamental latency limitations of today's prominent centralized cloud systems [1]. These architectures are expected to play an important role also in next-generation tactical networks for supporting both computation-intensive and latency-sensitive services in the tactical field [2].

With MEC, services can be housed in cellular base stations (BSs), or even mobile nodes acting as access points (APs) for other nodes in the tactical terrain, that are endowed with computing capabilities (edge servers) and which can be used to accommodate service requests from users lying in their coverage regions. The computation capacity of BSs and APs, however, is much more limited than that of centralized clouds, and may not suffice to satisfy all user requests. This naturally raises the question of *where to execute* each service so as to better reap the benefits of available computation resources to serve as many requests as possible.

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

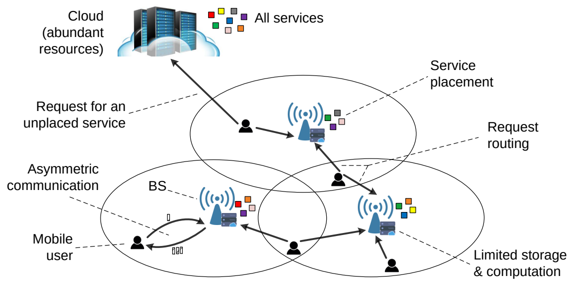

Cloud (abundant resources)
All services
Service placement
Request for an unplaced service
Request routing
Asymmetric communication
BS
Mobile user
Limited storage & computation

Fig. 1: An example MEC system. Service placement and request routing are constrained by the storage, computation, and bandwidth resources of BSs.

While there have been several interesting approaches to determine the execution (or offloading) of services in MEC, e.g., [3] and [4], to cite two of the most recent, an important aspect has been hitherto overlooked. Specifically, many services today require not only the allocation of computation resources, but also a *non-trivial amount of data that needs to be pre-stored* (or pre-placed) at the edge server. In an Augmented Reality (AR) service, for example, the placement of the object database and the visual recognition models is needed in order to run classification or object recognition before delivering the augmented information to the user (e.g., display battlefield information on the AR headset of a soldier) [5]. Yet, the storage capacity of edge servers may not be large enough to support all offered services.

The above issue is further complicated by the complex communication requirements of modern tactical services. Many of these services require uploading data from the user to be used as input for service execution, whose output must then be downloaded for consumption by the user. Such *bidirectional communication* may be asymmetric in general, taking up different portions of edge server's uplink and downlink bandwidth capacities [6].

In addition, the density of BSs (and edge servers) will be much higher in the anticipated 5G deployments [7] while such deployments are expected to be largely used in tactical operations. This is creating a *complex multi-cell environment* with users concurrently in range of multiple edge servers with overlapping coverage regions, and where the operator (e.g., tactical command center) can use multiple paths to route associated service requests. Figure 1 illustrates an example of such a system.

Evidently, in this context, MEC operators have a large repertoire of service placement and routing alternatives for satisfying the user requests. In order to serve as many requests as possible from the edge servers, the operator has to *jointly* optimize these decisions while simultaneously satisfying storage, computation, and communication constraints. Clearly, this is an important problem that differs substantially from previous related studies (e.g., see [3], [4]) that did not consider storage-constrained edge servers and asymmetric communication requirements. While a few recent works [8], [9], [10] studied the impact of storage in MEC, they neither considered all the features of these systems discussed above nor provided *optimal* or *approximate* solutions for the joint service placement and request routing problem.

B. Methodology and Contributions

In this paper, we follow a systematic methodology in order to address the above problem, summarized as follows.

1) We formulate the joint service placement and request routing problem (JSPRR) in multi-cell MEC networks aiming to minimize the load of the centralized cloud.
2) We identify several placement and routing problems in literature that are special cases of JSPRR, gaining insights into the complexity of the original problem.
3) Using a randomized rounding technique [11], we develop a bi-criteria algorithm that provably achieves approximation guarantees while violating the resource constraints in a bounded way.
4) We carry out evaluations to demonstrate the performance of the proposed algorithm. We show that, in many practical scenarios, our algorithm performs close-to-optimal and far better than a state-of-the-art method which neglects computation and bandwidth constraints.

The rest of the paper is organized as follows. Section II describes the system model and defines the JSPRR problem. We analyze the complexity of JSPRR and present approximation algorithms in Section III and IV, respectively. Section V presents our evaluations and Section VI concludes our work.

II. Model and Problem Definition

We consider a MEC system consisting of a set $\mathcal{N}$ of N BSs or APs in the tactical terrain equipped with storage, computation, and communication capabilities, and a set $\mathcal{U}$ of U mobile users, subscribers of the MEC operator, as depicted in Figure 1. The users may be arbitrarily distributed over the (possibly overlapping) coverage regions of the BSs, where $\mathcal{N}_u \subseteq \mathcal{N}$ denotes the set of BSs covering user u.

We consider multiple types of resources for the MEC BSs. First, each BS n has *storage capacity* R_n (hard disk) that can be used to pre-store data associated with services. Second, BS n has a CPU of *computation capacity* (i.e., maximum frequency) C_n that can be used to execute services in an on-demand manner. Third, BS n has uplink (downlink) *bandwidth capacity* $B_n^{\uparrow}$ ($B_n^{\downarrow}$) that can be used to upload (download) data from (to) mobile users requesting services.

The system offers a library $\mathcal{S}$ of S latency-sensitive services to the mobile users. Examples include augmented reality, video streaming and networked gaming. Services may have different requirements in terms of storage, CPU cycles, and uplink/downlink bandwidth resources. We denote by r_s the storage space occupied by the data associated with service s. The notation c_s indicates the required computation, while $b_s^{\uparrow}$ and $b_s^{\downarrow}$ indicate the uplink and downlink bandwidth required to satisfy a request for service s, respectively.

The system receives service requests from the users in a stochastic manner. Without loss of generality, we assume that each user u performs one request for a service denoted by s_u[1]. User requests can be predicted for a certain time period by using learning techniques [8].

The request of user u can be routed to a nearby BS in $\mathcal{N}_u$ provided that service s_u is locally stored and the BS has enough computation and bandwidth resources. If there is no such BS, we assume that the user can access the centralized cloud, which serves as a last resort for all users. Accessing the cloud, however, may cause high delay due to its long distance from the users, and therefore should be avoided.

The network operator needs to decide in which BSs to place the services and how to route user requests to them. To model these decisions, we introduce two sets of optimization variables: (i) $x_{ns} \in \{0,1\}$ which indicates whether service s is placed in BS n ($x_{ns} = 1$) or not ($x_{ns} = 0$), and (ii) $y_{nu} \in \{0,1\}$ which indicates whether the request of user u is routed to BS n ($y_{nu} = 1$) or not ($y_{nu} = 0$). Similarly, we denote by y_{lu} the decision to route the request of user u to the (centralized) cloud. We refer by *service placement* and *request routing* policies to the respective vectors:

$$\boldsymbol{x} = (x_{ns} \in \{0,1\} \ : n \in \mathcal{N}, s \in \mathcal{S}) \tag{1}$$

$$\boldsymbol{y} = (y_{nu} \in \{0,1\} \ : n \in \mathcal{N} \cup \{l\}, u \in \mathcal{U}) \tag{2}$$

The service placement and request routing policies need to satisfy several constraints. First, each user request needs to be routed to exactly one of the nearby BSs, or the cloud:

$$\sum_{n \in \mathcal{N}_u \cup \{l\}} y_{nu} = 1, \ \forall u \in \mathcal{U} \tag{3}$$

Second, in order to route the request of user u to BS n, service s_u must be placed in BS n:

$$y_{nu} \leq x_{ns_u}, \ \forall n \in \mathcal{N}, u \in \mathcal{U} \tag{4}$$

Third, the total amount of service data placed in a BS must not exceed its storage capacity:

$$\sum_{s \in \mathcal{S}} x_{ns} r_s \leq R_n, \ \forall n \in \mathcal{N} \tag{5}$$

Fourth, the total computation load generated by the user requests routed to BS n must not exceed its computation capacity:

$$\sum_{u \in \mathcal{U}} y_{nu} c_{s_u} \leq C_n, \ \forall n \in \mathcal{N} \tag{6}$$

[1]If a user performs multiple requests, we can split it into multiple users.

Fifth, the total bandwidth load generated by the requests routed to BS n must not exceed its uplink and downlink bandwidth capacity:

$$\sum_{u\in\mathcal{U}} y_{nu} b^{\uparrow}_{s_u} \leq B^{\uparrow}_{n},\ \forall n \in \mathcal{N} \quad (7)$$

$$\sum_{u\in\mathcal{U}} y_{nu} b^{\downarrow}_{s_u} \leq B^{\downarrow}_{n},\ \forall n \in \mathcal{N} \quad (8)$$

The goal of the network operator is to find the joint service placement and request routing policy that maximizes the number of requests served by the BSs, or, equivalently, minimizes the load of the cloud:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \textstyle\sum_{u\in\mathcal{U}} y_{lu} \qquad (9)\\ \text{s.t.} \quad & \text{constraints: } (1) - (8) \end{aligned}$$

We refer by *JSPRR* to the above problem. In the next two sections, we analyze the complexity of this problem and propose approximation algorithms.

III. Complexity Analysis

The JSPRR problem is *NP-Hard* since it generalizes the knapsack problem [12] by comprising multiple packing constraints (inequalities (5)-(8)). The problem remains challenging even when simplified by the assumption of *homogeneous (unit-sized) service requirements*, i.e., $r_s = c_s = b^{\uparrow}_{s} = b^{\downarrow}_{s} = 1$ $\forall s$. Interestingly, this simplified problem includes as special cases several *well-studied placement and routing problems* in literature, which shows the universality of our model.

A. Special case 1: Non-overlapping BS coverage regions

In the first special case, we make the simplifying assumption (in addition to the homogeneity of service requirements) that the coverage regions of the BSs do not overlap with each other. This particularly applies to sparse BS deployments where the BSs are located far away one from the other. It follows that the JSPRR problem can be *decomposed into N independent subproblems*, one per BS n. The objective of subproblem n is to maximize the number of requests served by BS n.

It is not difficult to show that there is always an optimal solution to subproblem n that places in BS n the R_n most locally popular services, i.e., the services requested by most users inside the coverage region of BS n. Then, BS n will admit as many requests for the placed services as its computation and bandwidth capacities C_n, $B^{\uparrow}_{n}$ and $B^{\downarrow}_{n}$ can handle. Therefore, JSPRR is trivial in this special case.

B. Special case 2: Data placement/caching problem

In the second special case, we allow the coverage regions of BSs to overlap, but we make the simplifying assumption that the computation and bandwidth resources are non-congestible, i.e., they always suffice to route all user requests to BSs.

Without the computation and bandwidth constraints, the placement and routing problem becomes much simpler. For a given service placement solution $\boldsymbol{x}$, finding the optimal request routing policy $\boldsymbol{y}$ is straightforward; simply route each user request to a nearby BS having stored the requested service, if any; otherwise to the cloud. This special case has been studied

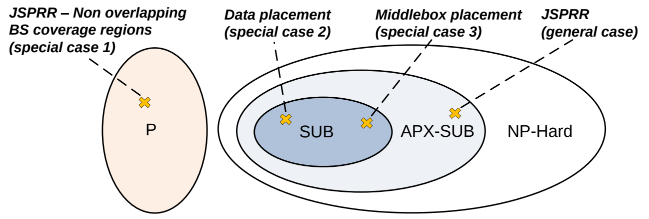

JSPRR – Non overlapping BS coverage regions (special case 1)
Data placement (special case 2)
Middlebox placement (special case 3)
JSPRR (general case)
P
SUB
APX-SUB
NP-Hard

Fig. 2: Complexity of special cases of JSPRR: Polynomial-time solvable (P), Submodular (SUB), and Approximately submodular (APX-SUB) classes.

in literature under the title *'data placement'* [13] or *'caching' problem* [14]. This problem asks to place data items (services) to caches (BSs) with the objective of maximizing the total number of requests served by the caches.

While the data placement problem is NP-Hard, several approximation algorithms are known in literature. The main method used to derive such approximations is based on showing the submodularity property of the optimization problem. That is, to show that the marginal value of the objective function never increases as more data items are placed in the caches. Having shown the submodularity property, several 'classic' algorithms can be applied, with the most known being greedy, local search, and pipage rounding [14]. Among the three algorithms, the greedy is the simplest and fastest, and, hence, the most practical.

C. Special case 3: Middlebox placement problem

In the third special case, we allow the coverage regions of BSs to overlap and the computation and bandwidth resources to be congestible, but we make the simplifying assumption that the storage capacities are unit-sized ($R_n = 1$ $\forall n$). That is, we assume that only one service can be stored per BS.

Under this special case, the JSPRR problem can be reduced to the *middlebox placement' problem* [15]. While there exist many different variants of the middlebox placement problem in literature, typically, this problem asks to pick m out of p nodes in a network to deploy middleboxes. The goal is to maximize the total number of source-destination flows (out of q flows) that can be routed through network paths containing at least one middlebox, subject to a constraint k that limits the number of flows processed by each middlebox.

Although the reduction is not straightforward, the main idea is to construct the middlebox placement instance by creating: (i) a distinct node for each pair of a BS and a service ($p = NS$ nodes in total) and (ii) a distinct flow for each user ($q = U$ flows in total). We then allow each flow to be routed through any node whose BS-service pair satisfies that the BS covers the respective user and the service is the one requested by that user. The question is which $m = N$ out of the $p = NS$ nodes to pick to deploy middleboxes.

Recent works have shown that the maximum flow objective of the middlebox problem is a submodular function [15]. Therefore, this problem can be solved by using the same approximation algorithms mentioned in special case 2.

D. General case: Non-submodular

Although it would be tempting to conjecture that our JSPRR problem is submodular in its general form (with overlapping coverage regions, congestible bandwidth and computation and large storage capacities), we can construct counter-examples where this property does not hold. First, we introduce the definition of submodular functions.

Definition 1. *Given a finite set of elements $\mathcal{G}$ (ground set), a function $f : 2^{\mathcal{G}} \to \mathbb{R}$ is submodular if for any sets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{G}$ and every element $g \notin \mathcal{B}$, it holds that:*

$$f(\mathcal{A} \cup \{g\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{g\}) - f(\mathcal{B}) \tag{10}$$

Next, we introduce the element e_{ns} to denote the placement of service s in BS n. The ground set is given by $\{e_{11}, \ldots, e_{NS}\}$. Every possible service placement policy can be expressed by a subset $\mathcal{E} \subseteq \mathcal{G}$ of elements, where the elements included in $\mathcal{E}$ correspond to the service placement. Given a service placement $\mathcal{E}$, we denote by $f(\mathcal{E})$ the maximum number of user requests that can be satisfied by the BSs.

We will construct a counter-example where the function $f(\mathcal{E})$ is not submodular. Specifically, we consider a system of $N = 2$ BSs and $U = 2$ users located in the intersection of the two coverage regions. The users request two different services denoted by s_1 and s_2. We set the computation capacities to $C_1 = C_2 = 1$ (i.e., at most one service request can be satisfied by each BS), while the storage and bandwidth capacities are abundant. The two placement sets we consider are $\mathcal{A} = \{e_{11}\}$ and $\mathcal{B} = \{e_{11}, e_{21}\}$, where $\mathcal{A} \subseteq \mathcal{B}$. Then, we can show that $f(\mathcal{B} \cup \{e_{12}\}) - f(\mathcal{B}) = 2 - 1 = 1 > 0 = 1 - 1 = f(\mathcal{A} \cup \{e_{12}\}) - f(\mathcal{A})$, which means that f is not submodular.

E. General case: Approximately-submodular

Although our JSPRR problem does not fall into the class of submodular problems, we can show that it belongs to the wider class of *approximately submodular* problems [16]. The complexity of JSPRR for the general and special cases is illustrated in Figure 2.

Definition 2. *A function $f : 2^{\mathcal{G}} \to \mathbb{R}$ is δ-approximately submodular if there exists a submodular function $F : 2^{\mathcal{G}} \to \mathbb{R}$ such that for any $\mathcal{E} \subseteq \mathcal{G}$:*

$$(1 - \delta)F(\mathcal{E}) \leq f(\mathcal{E}) \leq (1 + \delta)F(\mathcal{E}) \tag{11}$$

We define by $F(\mathcal{E})$ the maximum number of user requests that can be satisfied by the BSs given the service placement set $\mathcal{E}$ in the special case that the bandwidth and computation resources are non-congestible (special case 2). Since there are fewer constraints in this special case than in the general case, it holds that $f(\mathcal{E}) \leq F(\mathcal{E})$. Therefore, for any $\delta \in [0, 1]$, we have $f(\mathcal{E}) \leq (1 + \delta)F(\mathcal{E})$. What remains to find is a δ value that satisfies the first inequality in (11), i.e., $(1 - \delta)F(\mathcal{E}) \leq f(\mathcal{E})$.

We note that when computing the value of $F(\mathcal{E})$, the BS n is allowed to satisfy all the requests for stored services generated by users in its coverage region. We denote by Φ_n the number of these requests. In case that it happens $\Phi_n \leq C_n$, $\Phi_n \leq B_n^{\uparrow}$ and $\Phi_n \leq B_n^{\downarrow}$ $\forall n$, then the computation and bandwidth resources are non-congestible and we have $f(\mathcal{E}) = F(\mathcal{E})$. In the other case that, for some n, it happens $\Phi_n > C_n$ or $\Phi_n > B_n^{\uparrow}$ or $\Phi_n > B_n^{\downarrow}$, then the BS n can process up to Φ_n / C_n times more requests, compared to $f(\mathcal{E})$. Similarly, the BS n can receive (deliver) data from (to) up to $\Phi_n / B_n^{\uparrow}$ ($\Phi_n / B_n^{\downarrow}$) times more users. Therefore, the total number of satisfied requests is upper bounded by:

$$F(\mathcal{E}) \leq \max_{n \in \mathcal{N}} \{\frac{\Phi_n}{C_n},\ \frac{\Phi_n}{B_n^{\uparrow}},\ \frac{\Phi_n}{B_n^{\downarrow}},\ 1\} f(\mathcal{E}) \tag{12}$$

where the value 1 inside the $\max$ operator ensures that $F(\mathcal{E})$ will never be lower than $f(\mathcal{E})$. We thus can ensure that $(1 - \delta)F(\mathcal{E}) \leq f(\mathcal{E})$ by picking:

$$\delta = 1 - \frac{1}{\max_{n \in \mathcal{N}} \{\frac{\Phi_n}{C_n},\ \frac{\Phi_n}{B_n^{\uparrow}},\ \frac{\Phi_n}{B_n^{\downarrow}},\ 1\}} \tag{13}$$

The problem of maximizing a δ-approximately submodular function has been studied in the past [16]. Based on the results in [16], we can use a simple greedy algorithm to achieve the approximation ratio described in the following proposition.

Proposition 1. *The Greedy algorithm returns a solution set $\mathcal{E}^*$ such that:*

$$f(\mathcal{E}^*) \geq \frac{1}{2}\Big(\frac{1 - \delta}{1 + \delta}\Big) \frac{1}{1 + \frac{\sum_{n \in \mathcal{N}} R_n \delta}{1 - \delta}} \max_{\mathcal{E}} f(\mathcal{E}) \tag{14}$$

The above approximation ratio worsens as the network becomes more congested (δ increases) and the storage capacities increase (R_n). This suggests that the JSPRR problem is substantially harder than the placement and routing problems described above, and thus *a method of different philosophy is needed to find a tight approximation ratio*.

IV. Approximation Algorithm

In this section, we present one of the main contributions of this work; a novel approximation algorithm for the JSPRR problem that leverages a *randomized rounding* technique and is referred to as Service Placement and Routing via Randomized Rounding, or SPR^3. The SPR^3 algorithm is described in detail below and summarized in Algorithm 1.

The SPR^3 algorithm starts by solving the linear relaxation of the JSPRR problem (line 1). That is, it relaxes the variables $\{x_{ns}\}$ and $\{y_{nu}\}$ to be fractional, rather than integer. The Linear Relaxation of JSPRR problem, *LR-JSPRR* for short, can be expressed as follows:

$$\min_{\boldsymbol{x}, \boldsymbol{y}} \quad \textstyle\sum_{u \in \mathcal{U}} y_{lu} \tag{15}$$

$$\text{s.t.} \quad \text{constraints: (3) } - \text{ (8)}$$

$$x_{ns} \in [0, 1],\ \forall n \in \mathcal{N}, s \in \mathcal{S} \tag{16}$$

$$y_{nu} \in [0, 1],\ \forall n \in \mathcal{N} \cup \{l\}, u \in \mathcal{U} \tag{17}$$

where we have replaced equations (1)-(2) with (16)-(17). Since the objective and the constraints of the above problem are linear, it can be optimally solved in polynomial time using a linear program solver. We denote by $\{x_{ns}^{\dagger}\}$ and $\{y_{nu}^{\dagger}\}$ the optimal solution values. The next step is to round these values to obtain an integer solution, denoted by $\{\widehat{x}_{ns}\}$ and $\{\widehat{y}_{nu}\}$. For each pair of node n and service s, the algorithm rounds variable $\widehat{x}_{ns}$ to 1 with probability $x_{ns}^{\dagger}$ (lines 2-3). Each rounding decision is taken independently from each other.

Algorithm 1: SPR^3 algorithm

```
1  Solve the linear relaxation of JSPRR problem to obtain ($\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger$) optimal solution.
2  for $n \in \mathcal{N}$, $s \in \mathcal{S}$ do
3      Set $\widehat{x}_{ns} = 1$ with probability $x^\dagger_{ns}$.
   end
4  for $u \in \mathcal{U}$ do
5      Define $\mathcal{N}'_u = (n \in \mathcal{N}_u \ : \ \widehat{x}_{ns_u} = 1)$.
6      if $\mathcal{N}'_u = \emptyset$ then
7          Set $\widehat{y}_{lu} = 1$ with probability $\Theta_u$.
       else
8          Set $\widehat{y}_{nu} = 1$, $n \in \mathcal{N}'_u$, with probability $\frac{y^\dagger_{nu}}{x^\dagger_{ns_u}}$,
9          and $\widehat{y}_{lu} = 1$ with probability $\left[\frac{y^\dagger_{lu} - \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})}{1 - \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})}\right]_+$
10         Among all $n \in \mathcal{N}'_u$ such that $\widehat{y}_{nu} = 1$, pick one of them uniformly at random.
11         if all $n \in \mathcal{N}'_u$ are such that $\widehat{y}_{nu} = 0$ then
12             Pick the cloud value $\widehat{y}_{lu}$.
           end
       end
   end
13 Output $\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}}$
```

Finally, the algorithm uses the rounded placement variables $\{\widehat{x}_{ns}\}$ to decide the rounding of the routing variables (lines 4-12). For each user u, it defines the set of nearby BSs that have stored the requested service s_u by $\mathcal{N}'_u$ (line 5) and uses this set to distinguish between two cases: (i) if user u cannot find service s_u in any of the nearby BSs, i.e., $\mathcal{N}'_u = \emptyset$, then the user request is routed to the cloud with probability Θ_u (lines 6-7) given by:

$$\Theta_u = \begin{cases} 1, & \text{if } y^\dagger_{lu} \geq \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u}) \\ \frac{y^\dagger_{lu}}{\prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})}, & \text{else} \end{cases} \tag{18}$$

(ii) otherwise, the user is randomly routed to one of the BSs in $\mathcal{N}'_u$ or the cloud (lines 8-12). The routing probabilities depend on the fractional values $\{x^\dagger_{ns}\}$ and $\{y^\dagger_{nu}\}$. Higher probability is given to BSs with larger $y^\dagger_{nu}$ values. If more than one of the $y^\dagger_{nu}$ values are rounded to 1, only one of them is picked uniformly at random. Routing to the cloud is considered only if none of the BS values is picked. The notation $[.]_+$ in line 9 of the algorithm denotes the ramp function, i.e., $[\alpha]_+ = \max\{a, 0\}$.

Subsequently, we provide guarantees on the quality of the solution returned by the SPR^3 algorithm. We begin with the following lemma.

Lemma 1. *The SPR^3 algorithm routes all user requests with high probability.*

Proof. For a given user u, there are two cases when rounding the fractional variable $y^\dagger_{nu}$ to $\widehat{y}_{nu}$ for a BS ($n \in \mathcal{N}$) or the cloud ($n = l$): (i) there is no nearby BS having stored the requested service ($\mathcal{N}'_u = \emptyset$) and (ii) there is at least one such BS ($\mathcal{N}'_u \neq \emptyset$). The probability that the request of user u is routed to the cloud is given by:

$$\begin{aligned} \Pr[\widehat{y}_{lu} = 1] &= \Pr\left[\widehat{y}_{lu} = 1 \mid \mathcal{N}'_u = \emptyset\right] \Pr\left[\mathcal{N}'_u = \emptyset\right] \\ &+ \Pr\left[\widehat{y}_{lu} = 1 \mid \mathcal{N}'_u \neq \emptyset\right] \Pr\left[\mathcal{N}'_u \neq \emptyset\right] \\ &= \Theta_u \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u}) \\ &+ \left[\frac{y^\dagger_{lu} - \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})}{1 - \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})}\right]_+ (1 - \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})) = y^\dagger_{lu} \end{aligned} \tag{19}$$

The first equation is by the definition of conditional probability. The second equation is by replacing the probability values in lines 7 and 9 of the algorithm and due to the fact that the $\{x^\dagger_{ns}\}$ variables are rounded independently of one another (hence, $\Pr[\mathcal{N}'_u = \emptyset] = \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})$). To show the third equation we consider two cases: (i) $y^\dagger_{lu} \geq \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})$ and (ii) $y^\dagger_{lu} < \prod_{n \in \mathcal{N}'_u}(1 - x^\dagger_{ns_u})$, and replace the values of Θ_v and $[.]_+$ accordingly. In both cases, the end result of the equation will be equal to $y^\dagger_{lu}$.

Similarly, the probability that the request of user u is routed to BS n is given by:

$$\begin{aligned} \Pr[\widehat{y}_{nu} = 1] &= \Pr\left[\widehat{y}_{nu} = 1 \mid \widehat{x}_{ns_u} = 1\right] \Pr\left[\widehat{x}_{ns_u} = 1\right] \\ &+ \Pr\left[\widehat{y}_{nu} = 1 \mid \widehat{x}_{ns_u} = 0\right] \Pr\left[\widehat{x}_{ns_u} = 0\right] \\ &= \frac{y^\dagger_{nu}}{x^\dagger_{ns_u}} x^\dagger_{ns_u} = y^\dagger_{nu} \end{aligned} \tag{20}$$

The first equation is by the definition of conditional probability and the fact that the rounding decision of $\widehat{y}_{nu}$ variable depends on the $\widehat{x}_{ns_u}$ value regardless of the $\mathcal{N}'_u$ set. The second equation is by replacing the probability value in line 8 of the algorithm and because $\Pr[\widehat{y}_{nu} = 1 \mid \widehat{x}_{ns_u} = 0] = 0$.

The sum of probabilities of routing the request of user u to the cloud or the BSs is given by:

$$\sum_{n \in \mathcal{N}_u \cup \{l\}} \Pr[\widehat{y}_{nu} = 1] = \sum_{n \in \mathcal{N}_u \cup \{l\}} y^\dagger_{nu} = 1 \tag{21}$$

where the first equality holds due to (19) and (20), and the second due to (3). The above is an upper bound on the probability of routing the request of user u except for an additive gap that converges to zero as the number of BSs covering user u and/or their capacities increase. □

By construction, SPR^3 routes requests only to BSs that have stored the respective service ($\mathcal{N}'_u$ set in line 5) or to the cloud. Therefore, constraint (4) is satisfied. Next, we study whether the remaining constraints in (5), (6), (7), and (8) are satisfied.

Lemma 2. *The solution returned by the SPR^3 algorithm satisfies in expectation the storage, computation, and bandwidth capacity constraints in (5), (6), (7), and (8).*

Proof. We begin with the storage capacity constraint. The expected amount of data placed in BS n is given by:

$$\mathbb{E}[\sum_{s\in\mathcal{S}} \widehat{x}_{ns} r_s] = \sum_{s\in\mathcal{S}} \Pr[\widehat{x}_{ns}=1] r_s = \sum_{s\in\mathcal{S}} x^{\dagger}_{ns} r_s = R_n \quad (22)$$

where the second equation is because the $\{\widehat{x}_{ns}\}$ variables are binary, with success probabilities the fractional values $\{x^{\dagger}_{ns}\}$. The last equation is due to constraint (5) and the fact that it would be wasteful to not use all the storage space.

Next, we consider the computation capacity constraint. The expected computation load of BS n is given by:

$$\mathbb{E}[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u}] = \sum_{u\in\mathcal{U}} \Pr[\widehat{y}_{nu}=1] c_{s_u} = \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u} \leq C_n \quad (23)$$

where the second equation holds due to equation (20). The inequality is by constraint (6). Similar inequalities can be shown for the uplink/downlink bandwidth constraints:

$$\mathbb{E}[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} b^{\uparrow}_{s_u}] = \sum_{u\in\mathcal{U}} \Pr[\widehat{y}_{nu}=1] b^{\uparrow}_{s_u} = \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} b^{\uparrow}_{s_u} \leq B^{\uparrow}_n \quad (24)$$

$$\mathbb{E}[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} b^{\downarrow}_{s_u}] = \sum_{u\in\mathcal{U}} \Pr[\widehat{y}_{nu}=1] b^{\downarrow}_{s_u} = \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} b^{\downarrow}_{s_u} \leq B^{\downarrow}_n \quad (25)$$

where we have used equations (20), (7), and (8). □

A similar result holds for the objective function value.

Lemma 3. *The objective value returned by the SPR^3 algorithm is in expectation equal to that of the optimal fractional solution.*

Proof. The expected number of user requests routed to the cloud by SPR^3 is given by:

$$\mathbb{E}[\sum_{u\in\mathcal{U}} \widehat{y}_{lu}] = \sum_{u\in\mathcal{U}} \Pr[\widehat{y}_{lu}=1] = \sum_{u\in\mathcal{U}} y^{\dagger}_{lu} \quad (26)$$

where the second equation holds due to equation (19). □

The above lemmas have shown that the SPR^3 algorithm satisfies *in expectation* all the constraints and achieves the optimal objective value. However, *in practice*, the constraints may be violated. Therefore, it is important to bound the factor by which this happens.

Theorem 1. *The amount of data placed by the SPR^3 algorithm in BS n will not exceed its storage capacity by a factor larger than $\frac{3\ln(S)}{R_n}+4$ with high probability.*

Proof. For a given BS n, the products $\widehat{x}_{ns} r_s\ \forall s \in \mathcal{S}$ are independent random variables with expected total value $\mathbb{E}[\sum_{s\in\mathcal{S}} \widehat{x}_{ns} r_s] = R_n$ (cf. equation (22)). Moreover, by appropriately normalizing the r_s and R_n values, we can ensure that the $\widehat{x}_{ns} r_s$ variables take values within $[0,1]$. Therefore, we can apply the Chernoff Bound theorem [17] to show that for any $\epsilon > 0$:

$$\Pr[\sum_{s\in\mathcal{S}} \widehat{x}_{ns} r_s \geq (1+\epsilon) R_n] \leq \exp^{\frac{-\epsilon^2 R_n}{2+\epsilon}} \quad (27)$$

Next, we find an ϵ value for which the probability upper bound above becomes very small. Specifically, we require that:

$$\exp^{\frac{-\epsilon^2 R_n}{2+\epsilon}} \leq \frac{1}{S^3} \quad (28)$$

which means that the probability bound goes quickly (at a cubic rate) to zero as the number of services increases. In order for this to be true, the ϵ value must satisfy:

$$\epsilon \geq \frac{3\ln(S)}{2R_n} + \sqrt{\frac{9\ln^2(S)}{4R_n^2} + \frac{6\ln(S)}{R_n}} \quad (29)$$

The above condition holds if we pick:

$$\epsilon = \frac{3\ln(S)}{R_n} + 3 \quad (30)$$

since, in practice, $R_n \geq \ln(S)$. Finally, we upper bound the probability that *any* of the BS storage capacities is violated:

$$\Pr[\bigcup_{n\in\mathcal{N}} \sum_{s\in\mathcal{S}} \widehat{x}_{ns} r_s \geq (1+\epsilon) R_n]$$
$$\leq \sum_{n\in\mathcal{N}} \Pr[\sum_{s\in\mathcal{S}} \widehat{x}_{ns} r_s \geq (1+\epsilon) R_n] \leq N \frac{1}{S^3} \leq \frac{1}{S^2} \quad (31)$$

where the first inequality is due to the Union Bound theorem. The second inequality is due to inequality (28) and because the number of BSs is N. The last inequality is because, in practice, the service library size is larger than the number of BSs ($S > N$). Therefore, with high probability, the storage capacity of any BS n will not be exceeded by more than a factor of $1+\epsilon = \frac{3\ln(S)}{R_n}+4$. □

Theorem 2. *The computation load of BS n returned by the SPR^3 algorithm will not exceed its capacity by more than a factor of $\frac{3\ln(S)}{\lambda^{\dagger}}+4$ with high probability, where $\lambda^{\dagger}$ is the minimum computation load among BSs in the optimal fractional solution.*

Proof. The proof is similar to Theorem 1. For a given BS n, the variables $\widehat{y}_{nu} c_{s_u}\ \forall u \in \mathcal{U}$ are independent with expected total value $\mathbb{E}[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u}] = \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}$ (cf. inequality (23)). Moreover, they can be normalized to take values within $[0,1]$. Therefore, we can apply the Chernoff Bound theorem:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon) \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}] \leq \exp^{\frac{-\epsilon^2 \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}}{2+\epsilon}} \quad (32)$$

Unlike storage, however, the expected computation load may not be equal to the capacity, i.e., $\sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u} \neq C_n$. Therefore, we cannot replace it in the above inequality. To overcome this obstacle, we use the fact that $\sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u} \leq C_n$ (by constraint (6)) and $\lambda^{\dagger} \leq \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}$ (by definition of $\lambda^{\dagger}$) to show the following two inequalities:

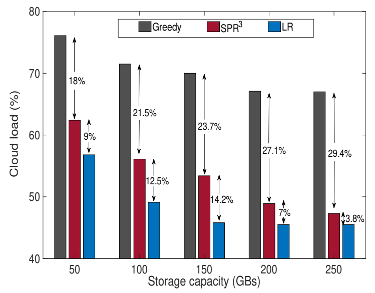
$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon) C_n] \leq \Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon) \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}] \quad (33)$$

$$\exp^{\frac{-\epsilon^2 \sum_{u\in\mathcal{U}} y^{\dagger}_{nu} c_{s_u}}{2+\epsilon}} \leq \exp^{\frac{-\epsilon^2 \lambda^{\dagger}}{2+\epsilon}} \quad (34)$$

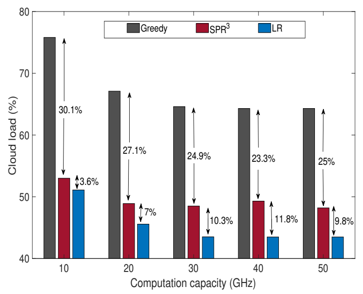
By combining inequalities (32), (33), and (34), we obtain:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon) C_n] \leq \exp^{\frac{-\epsilon^2 \lambda^{\dagger}}{2+\epsilon}} \quad (35)$$

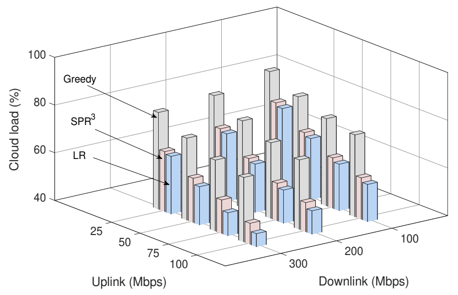
To complete the proof, we will find an ϵ value for which the probability upper bound above becomes very small, i.e., at



(a) Impact of storage capacity.



(b) Impact of computation capacity.



(c) Impact of bandwidth capacities.

Fig. 3: Cloud load for different (a) storage, (b) computation, and (c) uplink/downlink bandwidth capacities of BSs.

most $1/S^3$. Similarly to Theorem 1, we can set $\epsilon = \frac{3\ln(S)}{\lambda^\dagger} + 3$. Then, we can upper bound the probability that *any* of the computation capacities is violated by:

$$\Pr[\bigcup_{n\in\mathcal{N}} \sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon)C_n]$$
$$\leq \sum_{n\in\mathcal{N}} \Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu} c_{s_u} \geq (1+\epsilon)C_n] \leq N\frac{1}{S^3} \leq \frac{1}{S^2} \quad (36)$$

This means that, with high probability, the computation capacity of any BS will not be exceeded by more than a factor of $1+\epsilon = \frac{3\ln(S)}{\lambda^\dagger} + 4$. □

Using similar arguments, the following two theorems can be proved for the uplink and downlink bandwidth capacities.

Theorem 3. *The uplink bandwidth load of BS n returned by the SPR^3 algorithm will not exceed its capacity by more than a factor of $\frac{3\ln(S)}{\mu^\dagger} + 4$ with high probability, where $\mu^\dagger$ is the minimum uplink bandwidth load among BSs in the optimal fractional solution.*

Theorem 4. *The downlink bandwidth load of BS n returned by the SPR^3 algorithm will not exceed its capacity by more than a factor of $\frac{3\ln(S)}{\nu^\dagger} + 4$ with high probability, where $\nu^\dagger$ is the minimum downlink bandwidth load among BSs in the optimal fractional solution.*

What remains it to describe the worst case performance of the (in expectation optimal) SPR^3 algorithm.

Theorem 5. *The objective value returned by the SPR^3 algorithm is at most $\frac{2\ln(S)}{\xi^\dagger} + 3$ times worse than the optimal with high probability, where $\xi^\dagger$ is the optimal objective value in the linear relaxed problem.*

V. Evaluation Results

In this section, we carry out evaluations to show the performance of the proposed SPR^3 algorithm. We consider a similar setup as in the previous work [8]. Here, $N = 9$ base stations (BSs) are regularly deployed on a grid network inside a 500m×500m area. $U = 1,000$ mobile users are distributed uniformly at random over the BS coverage regions (each of 150m radius). Each user requests one latency-sensitive service drawn from a library of $S = 1,000$ services. The service popularity follows the Zipf distribution with shape parameter 0.8. For each BS n, we set the storage capacity to $R_n = 200$ GBs, the computation capacity to $C_n = 20$ GHz and the uplink (downlink) bandwidth capacity to $B_n^\uparrow = 100$ ($B_n^\downarrow = 250$) Mbps. Yet, all these values are varied during the evaluations.

We set the resource requirements r_s, c_s, $b_s^\uparrow$ and $b_s^\downarrow$ of the $S = 1,000$ services randomly by mapping them to 4 real latency-sensitive services, namely Video streaming (VS), Face recognition (FR), Gzip (compression) and Augmented reality (AR). **Video streaming** requires significant storage (1GB - 10GB) and downlink rate (1Mbps - 25Mbps) capturing videos of various lengths and playback qualities. The computation and uplink rate requirements are negligible for this particular service. **Face recognition** consumes notable uplink bandwidth for video frame uploading (1Mbps to 8Mbps). It also consumes significant computation (up to 3GHz) and storage (at least 2 GB) for matching to a database of possibly thousands of frames, but the downlink rate is negligible [18]. **Gzip** generates downlink rate 4 times lower than uplink rate representing a compression ratio of 4. The computation is set within [0.04,0.32] GHz [19] while the storage footprint is small (20MB). **Augmented reality** is the most resource demanding service. It requires significant bandwidth for the upload of video frames and the download of holograms to be augmented to the frames. The hologram sizes are set to 1/4 of the original frames and hence the required downlink rate is lower. The computation is set similarly to the FR service while the storage can be more than 10 GBs [20].

We compare our algorithm with two baseline methods.

1) *Linear-Relaxation (LR):* The optimal (fractional) solution to the linear relaxation of JSPRR problem. This solution is found by running a linear solver and provides a lower bound to the optimal integer solution value.
2) *Greedy [14]:* Iteratively, places a service to a BS cache that reduces cloud load the most, until all caches are filled. Each request is routed to the nearest BS with the service, neglecting computation and bandwidth limits.

We first explore the impact of storage capacity R_n $\forall n$ on the load of the centralized cloud. In Figure 3a, R_n spans a wide range of values, starting from 50GBs to 250GBs. As expected, increasing storage capacities reduces cloud load for all the algorithms as more requests can be satisfied locally (offloaded) by the BSs. *The proposed SPR^3 algorithm performs significantly better than Greedy* with gains up to 29.4%

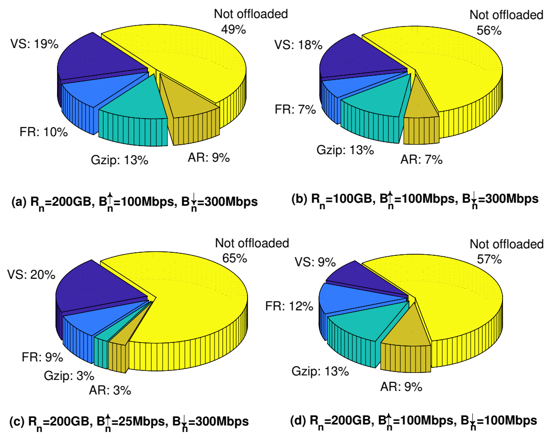


Fig. 4: Distribution of requests across services for different storage, uplink and bandwidth capacities: (a) default values, (b) reduced storage, (c) reduced uplink, (d) reduced downlink.

for $R_n = 250$GBs. At the same time, *the gap from LR, and hence optimal, is small* (no more than 14.2% gap) showing the efficiency of the proposed algorithm.

Next, we show the impact of computation capacity C_n in Figure 3b. While the cloud load reduces with C_n for all the algorithms, SPR^3 performs consistently better than Greedy and very close to LR. Especially when C_n is equal to 10GHz, the gains from Greedy climb up to 30.1% and the gap from LR is only 3.6%. Similarly, Figure 3c depicts the cloud load for different combinations of uplink ($B_n^{\uparrow}$) and downlink ($B_n^{\downarrow}$) bandwidth capacities. While the cloud load reduces with each of the $B_n^{\uparrow}$ and $B_n^{\downarrow}$ values for all the algorithms, SPR^3 achieves gains between 13.9% and 27.1% over Greedy. The gap from LR is no more than 8.7% in all combinations.

It is worth exploring which types of service requests are offloaded to the BSs and which are handled by the centralized cloud when the SPR^3 algorithm is applied. One might expect that roughly the same number of requests should be offloaded for all the types of services. Another might argue that resource demanding services such as AR should be offloaded less often than others in order to maximize the aggregate of offloaded requests (objective function). To shed light on this issue, Figure 4 depicts the distribution of requests across the four types of services (VS, FR, Gzip and AR) in four different scenarios. The values of storage, uplink and downlink bandwidth capacities are varied in each scenario. Subfigure (a) shows the results for the default values of $R_n = 200$GB, $B_n^{\uparrow} = 100$Mbps and $B_n^{\downarrow} = 300$Mbps. While about half of the requests are offloaded to the BSs, most of them are for video streaming since this type of service does not require any computation and uplink bandwidth which are the bottleneck resources. When we reduce the storage capacities from 200 to 100 GB (subfigure (b)), the volume of offloaded requests decreases for all the services but Gzip. This is because Gzip has almost zero storage footprint and therefore is not affected by alterations in this resource. Similarly, in subfigure (c), the reduction of the uplink rate from 100 to 25 Mbps reduces the offloaded requests for all services but the video streaming since the latter is the only service without any upload data requirements. We explore how the distribution changes when the downlink rate reduces from 300 to 100 Mbps in subfigure (d). This time the service that is affected the most is video streaming, with its share reducing from 19% to 9%.

VI. Conclusion

In this paper, we studied service placement and request routing in MEC-enabled networks with multiple routing path alternatives and multidimensional resource requirements. We showed that this problem generalizes well-known problems in literature that only consider a subset of resources, and is particularly relevant for next-generation data, computation, and communication intensive services. Using a randomized rounding technique, we proposed an algorithm that achieves provably close-to-optimal performance, which, to the best of our knowledge, is the first approximation for this problem.

References

[1] P. Mach, Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading'", *IEEE Communications Surveys & Tutorials., vol. 19, no. 3, pp. 1628-1656*, 2017.

[2] N. Bau, S. Endres, M. Gerz, F. Gokgoz, "A Cloud-based Architecture for an Interoperable, Resilient, and Scalable C2 Information System", *ICMCIS*, 2018.

[3] M. Chen, Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network", *IEEE Journal on Selected Areas in Communications, vol 36, no. 3, pp. 587-597*, 2018.

[4] H. Guo, J. Liu, J. Zhang, W. Sun, N. Kato, "Mobile-Edge Computation Offloading for Ultra-Dense IoT Networks", *IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4977-4988*, 2018.

[5] P. Jain, J. Manweiler, R. R. Choudhury, "Low Bandwidth Offload for Mobile AR", *ACM CoNEXT*, 2016.

[6] M.S. Elbamby, C. Perfecto, M. Bennis, K. Doppler,"Towards Low-Latency and Ultra-Reliable Virtual Reality", *IEEE Network*, 2018.

[7] X. Ge, S. Tu, G. Mao, C. X. Wang, T. Han, "5G Ultra-Dense Cellular Networks", *IEEE Wireless Communications, vol. 23, no. 1*, 2016.

[8] J. Xu, L. Chen, P. Zhou, "Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks", *IEEE Infocom*, 2018.

[9] T. He, H. Khamfroush, S. Wang, T.L. Porta, S. Stein, "It's Hard to Share: Joint Service Placement and Request Scheduling in Edge Clouds with Sharable and Non-sharable Resources", *IEEE ICDCS*, 2018.

[10] M. Chen, Y. Hao, L. Hu, M.S. Hossain, A. Ghoneim, "Edge-CoCaCo: Toward Joint Optimization of Computation, Caching, and Communication on Edge Cloud", *IEEE Wireless Communications, vol. 25, no. 3, pp. 21-27*, 2018.

[11] A. Srinivasan, "Approximation Algorithms Via Randomized Rounding: A Survey", *Advanced Topics in Mathematics, PWN, pp. 9-71*, 1999.

[12] H. Kellerer, U. Pferschy, D. Pisinger, "Knapsack Problems", *Springer*, 2004.

[13] I. Baev, R. Rajaraman, C. Swamy, "Approximation Algorithms for Data Placement Problems", *SIAM Journal on Comp., vol. 38*, 2008.

[14] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch and G. Caire, "FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers", *IEEE Transactions on Information Theory, vol. 59, no. 12*, 2013.

[15] T. Lukovszki, M Rost, S Schmid, "Approximate and Incremental Network Function Placement", *Journal of Parallel and Distributed Computing, vol. 120, pp. 159-169*, 2018.

[16] T. Horel, Y. Singer, "Maximization of Approximately Submodular Functions", *NIPS*, 2016.

[17] M. Mitzenmacher, E. Upfal, "Probability and Computing : Randomized Algorithms and Probabilistic Analysis", *Cambridge Univ. Press*, 2005.

[18] SmartFace, Plug & Play Face Recognition, https://www.innovatrics.com/face-recognition-solutions

[19] T. Q. Dinh, Q. D. La, T. Q. Quek, H. Shin, "Distributed Learning for Computation Offloading in Mobile Edge Computing", *IEEE Transactions on Communications, vol. 66, no. 12, pp. 6353-6367*, 2018.

[20] HP Windows Mixed Reality Headset Developer Edition, https://store.hp.com/us/en/cv/mixed-reality-headset