Service Placement and Request Routing in MEC Networks with Storage, Computation, and Communication Constraints

Konstantinos Poularakis[1], Jaime Llorca[2], Antonia M. Tulino[2,3], Ian Taylor[4], and Leandros Tassiulas[1]

[1]Department of Electrical Engineering and Institute for Network Science, Yale University, USA
[2]Nokia Bell Labs, New Jersey, USA
[3]Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Italy
[4]School of Computer Science and Informatics, Cardiff University, UK

***Abstract*—The proliferation of innovative mobile services such as augmented reality, networked gaming, and autonomous driving has spurred a growing need for low-latency access to computing resources that cannot be met 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 at the network edge, in proximity 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 dense 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, and 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 [2]. This trend is expected to continue unabated and play an important role in next-generation 5G networks for supporting both computation-intensive and latency-sensitive services [3].

Part of this work appeared in the proceedings of the IEEE International Conference on Computer Communications (Infocom), 2019 [1]. This publication was supported partly by the National Science Foundation under Grants CNS 1815676 and 1619129, the Army Research Office under Agreement Number W911NF18-10-378, and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

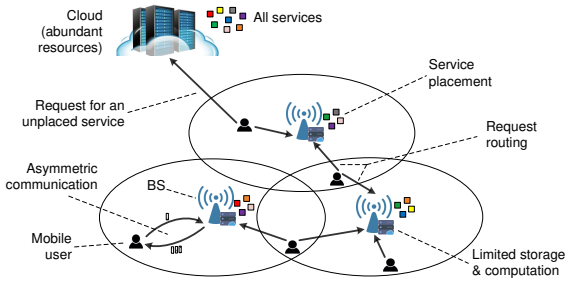


Fig. 1: An example MEC system. Service placement and request routing are constrained by the storage, computation, and bandwidth resources of BSs.

With MEC, services can be housed in base stations (BSs) (or edge servers close to BSs) endowed with computing capabilities that can be used to accommodate service requests from users lying in their coverage regions. The computation capacity of BSs, 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.

While there have been several interesting approaches to determine the execution (or offloading) of services in MEC, e.g., [4] and [5], 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 BS. 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 [6]. Yet, the storage capacity of BSs may not be large enough to support all offered services.

The above issue is further complicated by the services' communication requirements. Many modern 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 BSs' uplink and downlink bandwidth capacities [7].

In addition, the density of BSs has been increasing and is expected to reach up to 50 BSs per km^2 in future 5G deployments [8]. This is creating a *complex multi-cell environment* with users concurrently in range of multiple BSs with overlapping coverage regions, and where the operator 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 BSs, 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 [4], [5] and the discussion of related works in Section VII) that did not consider storage-constrained BSs and asymmetric communication requirements. While a few recent works [9], [10], [11] 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.

Given the above issues, the key open questions are:

- *Which services to place in each BS to best utilize their available storage capacity?*
- *How to route user requests to BSs without overwhelming their available computation and (uplink/downlink) bandwidth capacities?*
- *How the above decisions can be optimized in a joint manner to offload the centralized cloud as much as possible?*

B. Methodology and Contributions

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

1) We formulate the joint service placement and request routing problem (JSPRR) in dense multi-cell MEC networks aiming to minimize the load of the centralized cloud. We consider practical features of these systems such as overlapping coverage regions of BSs and multi-dimensional (storage, computation, and communication) resource constraints.
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 [12], we develop a multi-criteria algorithm that provably achieves approximation guarantees while violating the resource constraints in a bounded way. To the best of our knowledge, this is the first approximation algorithm for this problem.
4) We extend the results for dynamic scenarios where the user demand profiles change with time, and show how to adapt the solution accordingly.
5) 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 formally. We analyze the complexity of JSPRR and present algorithms with approximation guarantees in Section III and IV, respectively. Section V discusses practical extensions of our approach, while Section VI presents our evaluation results. We review our contribution compared to related works in Section VII and conclude our work in Section VIII.

II. Model and Problem Definition

We consider a MEC system consisting of a set $\mathcal{N}$ of $N \in \mathbb{N}$ BSs equipped with storage, computation, and communication capabilities, and a set $\mathcal{U}$ of $U \in \mathbb{N}$ 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 $S \in \mathbb{N}$ 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. If a user performs multiple requests, we can split it into multiple users. User requests can be predicted for a certain time period (e.g., a few hours) by using learning techniques [9]. Yet, user demand can change after that period as users may gain or lose interest in some services. We provide more details about this issue in Sections V and VI.

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, requests cannot be routed to BSs that are not nearby:

$$y_{nu} = 0, \ \forall u \in \mathcal{U}, \ n \notin \mathcal{N}_u \tag{4}$$

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

$$y_{nu} \leq x_{ns_u}, \ \forall n \in \mathcal{N}, u \in \mathcal{U} \tag{5}$$

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

$$\sum_{s \in \mathcal{S}} x_{ns} r_s \leq R_n, \ \forall n \in \mathcal{N} \tag{6}$$

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

$$\sum_{u \in \mathcal{U}} y_{nu} c_{s_u} \leq C_n, \ \forall n \in \mathcal{N} \tag{7}$$

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

$$\sum_{u \in \mathcal{U}} y_{nu} b^{\uparrow}_{s_u} \leq B^{\uparrow}_n, \ \forall n \in \mathcal{N} \tag{8}$$

$$\sum_{u \in \mathcal{U}} y_{nu} b^{\downarrow}_{s_u} \leq B^{\downarrow}_n, \ \forall n \in \mathcal{N} \tag{9}$$

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:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \textstyle\sum_{u \in \mathcal{U}} y_{lu} \tag{10}$$

$$\text{s.t.} \quad \text{Constraints: } (1) \ - \ (9)$$

We refer by *JSPRR* to the above problem. This is an integer optimization problem and such problems are typically challenging to solve. In the next sections, we discuss the relation to other known problems and propose approximation algorithms.

III. Relation to Known Problems

The JSPRR problem is *NP-Hard* since it generalizes the knapsack problem by comprising multiple packing constraints (Inequalities (6)-(9)). In this section, we investigate several special cases of the problem and show how these can be solved by making connections to some other *well-studied placement and routing problems* in literature. All the special cases we present make the simplifying assumption of homogeneous service requirements ($r_s = c_s = b^{\uparrow}_s = b^{\downarrow}_s = 1$ for all $s \in \mathcal{S}$), while each special case makes its own extra assumptions.

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, i.e., $\min\{C_n, B^{\uparrow}_n, B^{\downarrow}_n\}$ requests at most. Indeed, consider a solution that places in BS n a service s_1 requested by fewer users inside the respective coverage region than another service s_2. Then, one could swap the two services in the placement solution and route the same number of requests to BS n without changing the objective function value. Therefore, the JSPRR problem is trivial to solve in this special case.

B. Special case 2: Non-congestible computation & bandwidth

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. In other words, we assume that the capacities C_n, $B^{\uparrow}_n$ and $B^{\downarrow}_n$ are greater than or equal to the demand of users, so that we can remove Constraints (7)-(9) from the problem formulation without affecting the optimal solution.

Without the computation and bandwidth constraints, the problem becomes much simpler. For a given service placement $\boldsymbol{x}$, 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 route it to the cloud. This special case has been extensively studied in literature under the title *'data placement'* [13] or *'caching' problem* [14], [15]. 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 [15]. Among the three algorithms, the greedy is the simplest and fastest, and, hence, the most practical.

C. Special case 3: Unit-sized storage capacities

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 ($R_n = 1$ for all $n \in \mathcal{N}$). That is, we assume that only 1 service can be stored per BS.

The simplified JSPRR problem can be reduced to the *'middlebox placement' problem* [21], [22]. 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 subject to the constraint that at most $k = \min\{C_n, B_n^{\uparrow}, B_n^{\downarrow}\}$ flows can be routed through each node corresponding to BS n and the additional constraint that only 1 out of the S nodes corresponding to BS n can be picked (representing the storage constraint $R_n = 1$). The picked node will determine which of the S services is placed at BS n.

Recent works have shown that the maximum flow objective of the middlebox problem is a submodular function [21], [22]. 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:*

$$f(\mathcal{A} \cup \{g\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{g\}) - f(\mathcal{B}) \qquad (11)$$

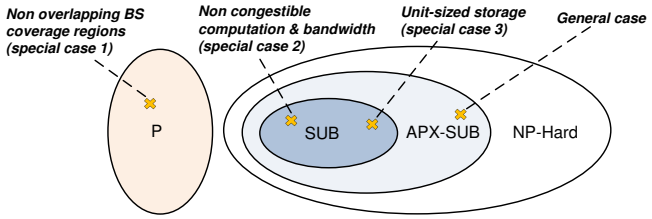


Fig. 2: Complexity of special cases of JSPRR: Polynomial-time solvable (P), Submodular (SUB), and Approximately submodular (APX-SUB) classes. All special cases are under the assumption of homogeneous service requirements.

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 $\mathcal{A} \subseteq \mathcal{B}$. We note that $f(\mathcal{A}) = f(\mathcal{B}) = 1$ since in both cases only one of the two services is stored (s_1), and hence only one of the two requests can be served. Besides, $f(\mathcal{A} \cup \{e_{12}\}) = 1$ since the computation constraint prevents BS 1 from serving both user requests. However, $f(\mathcal{B} \cup \{e_{12}\}) = 2$ since now each BS can serve one user request. Therefore, the marginal performance is larger for the set $\mathcal{B}$ than the $\mathcal{A}$, i.e., $f(\mathcal{B} \cup \{e_{12}\}) - f(\mathcal{B}) > f(\mathcal{A} \cup \{e_{12}\}) - f(\mathcal{A})$, which means that f is not submodular.

E. General case: Approximately-submodular

Although JSPRR does not fall into the class of submodular problems, we can show that it belongs to the wider class of *approximately submodular* problems [24]. The complexity of JSPRR for general and special cases is depicted 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 $\mathcal{E} \subseteq \mathcal{G}$:*

$$(1 - \delta)F(\mathcal{E}) \leq f(\mathcal{E}) \leq (1 + \delta)F(\mathcal{E}) \qquad (12)$$

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 (12), 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 $\Phi_n \leq B_n^{\downarrow}$ for all $n \in \mathcal{N}$, 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 $n \in \mathcal{N}$, 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:

$$F(\mathcal{E}) \leq \max_{n\in\mathcal{N}}\{\frac{\Phi_n}{C_n},\ \frac{\Phi_n}{B_n^{\uparrow}},\ \frac{\Phi_n}{B_n^{\downarrow}},\ 1\}f(\mathcal{E}) \tag{13}$$

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:

$$\delta = 1 - \frac{1}{\max_{n\in\mathcal{N}}\{\frac{\Phi_n}{C_n},\ \frac{\Phi_n}{B_n^{\uparrow}},\ \frac{\Phi_n}{B_n^{\downarrow}},\ 1\}} \tag{14}$$

The problem of maximizing a δ-approximately submodular function has been studied in the past [24]. Based on the results in [24], 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:*

$$f(\mathcal{E}^*) \geq \frac{1}{2}\Big(\frac{1-\delta}{1+\delta}\Big)\frac{1}{1+\frac{\sum_{n\in\mathcal{N}} R_n\delta}{1-\delta}}\max_{\mathcal{E}} f(\mathcal{E}) \tag{15}$$

Consider for example the case that the demand exceeds the available resources by up to 50%, i.e., there exists a BS n for which $\Phi_n = 1.5C_n$ or $\Phi_n = 1.5B_n^{\uparrow}$ or $\Phi_n = 1.5B_n^{\downarrow}$. Then, $\delta = 1/3$, and the approximation factor becomes:

$$f(\mathcal{E}^*) \geq \frac{1}{4}\frac{1}{1+\frac{\sum_{n\in\mathcal{N}} R_n}{2}}\max_{\mathcal{E}} f(\mathcal{E}) \tag{16}$$

The above approximation ratio worsens as the network becomes congested (δ increases) and the storage capacities increase (R_n). To find a tighter approximation, we present in next section another method based on randomized rounding.

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

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 Θ_u given in (20).
| | **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 $\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}}$

Linear Relaxation of JSPRR problem, *LR-JSPRR* for short, can be expressed as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \sum_{u\in\mathcal{U}} y_{lu} \tag{17}$$

$$\text{s.t.} \quad \text{Constraints: (3) } - \text{ (9)}$$

$$x_{ns} \in [0,1],\ \forall n \in \mathcal{N}, s \in \mathcal{S} \tag{18}$$

$$y_{nu} \in [0,1],\ \forall n \in \mathcal{N} \cup \{l\}, u \in \mathcal{U} \tag{19}$$

where we have replaced Equations (1)-(2) with (18)-(19). 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 [25]. We denote by $\{x^\dagger_{ns}\}$ and $\{y^\dagger_{nu}\}$ 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 $x^\dagger_{ns}$ (Lines 2-3). Each rounding decision is taken independently from each other.

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:

$$\Theta_u = \begin{cases} 1, & \text{if } y^\dagger_{lu} \geq \prod_{n\in\mathcal{N}_u}(1-x^\dagger_{ns_u}) \\ \frac{y^\dagger_{lu}}{\prod_{n\in\mathcal{N}_u}(1-x^\dagger_{ns_u})}, & \text{else} \end{cases} \tag{20}$$

(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 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. *SPR^3 algorithm routes every user request with high probability as the BS density and service requirements grow.*

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:

$$\begin{aligned}\Pr[\widehat{y}_{lu} = 1] &= \Pr\left[\widehat{y}_{lu} = 1 \mid \mathcal{N}'_u = \emptyset\right] \Pr\left[\mathcal{N}'_u = \emptyset\right] \\ &+ \Pr\left[\widehat{y}_{lu} = 1 \mid \mathcal{N}'_u \neq \emptyset\right] \Pr\left[\mathcal{N}'_u \neq \emptyset\right] \\ &= \Theta_u \prod_{n\in\mathcal{N}_u} (1 - x^\dagger_{ns_u}) \\ &+ \left[\frac{y^\dagger_{lu} - \prod_{n\in\mathcal{N}_u}(1 - x^\dagger_{ns_u})}{1 - \prod_{n\in\mathcal{N}_u}(1 - x^\dagger_{ns_u})}\right]_+ (1 - \prod_{n\in\mathcal{N}_u} (1 - x^\dagger_{ns_u})) \\ &= y^\dagger_{lu} \end{aligned} \tag{21}$$

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:

$$\begin{aligned}\Pr[\widehat{y}_{nu} = 1] &= \Pr\left[\widehat{y}_{nu} = 1 \mid \widehat{x}_{ns_u} = 1\right] \Pr\left[\widehat{x}_{ns_u} = 1\right] \\ &+ \Pr\left[\widehat{y}_{nu} = 1 \mid \widehat{x}_{ns_u} = 0\right] \Pr\left[\widehat{x}_{ns_u} = 0\right] \\ &= \frac{y^\dagger_{nu}}{x^\dagger_{ns_u}} x^\dagger_{ns_u} = y^\dagger_{nu} \end{aligned} \tag{22}$$

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:

$$\sum_{n\in\mathcal{N}_u\cup\{l\}} \Pr[\widehat{y}_{nu} = 1] = \sum_{n\in\mathcal{N}_u\cup\{l\}} y^\dagger_{nu} = 1 \tag{23}$$

where the first equation holds due to Equations (21) and (22), 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 goes to zero with BS density. Specifically, the probability that the request of user u is *not* routed by SPR^3 is given by:

$$\begin{aligned}&\Pr\Big[\sum_{n\in\mathcal{N}_u\cup\{l\}} \widehat{y}_{nu} = 0\Big] = \\ &= \Pr\Big[\sum_{n\in\mathcal{N}_u\cup\{l\}} \widehat{y}_{nu} = 0 \mid \mathcal{N}'_u = \emptyset\Big] \Pr\Big[\mathcal{N}'_u = \emptyset\Big] \\ &+ \Pr\Big[\sum_{n\in\mathcal{N}_u\cup\{l\}} \widehat{y}_{nu} = 0 \mid \mathcal{N}'_u \neq \emptyset\Big] \Pr\Big[\mathcal{N}'_u \neq \emptyset\Big] \end{aligned} \tag{24}$$

Furthermore, the probability that no nearby BS has stored s_u (in the rounded solution) can be bounded as:

$$\begin{aligned}\Pr[\mathcal{N}'_u = \emptyset] &= \Pr[\bigcap_{n\in\mathcal{N}_u} \{\widehat{x}_{ns_u} = 0\}] \\ &= \prod_{n\in\mathcal{N}_u} (1 - x^\dagger_{ns_u}) \\ &\leq (1 - \min_{\widetilde{\mathcal{N}}_u} x^\dagger_{ns_u})^{|\widetilde{\mathcal{N}}_u|} \end{aligned} \tag{25}$$

where $\widetilde{\mathcal{N}}_u = (n \in \mathcal{N}_u \ : \ x^\dagger_{ns_u} > 0)$ is the set of nearby BSs storing s_u in the fractional solution.

As BS density and service resource requirements increase then $|\widetilde{\mathcal{N}}_u|$ also grows, and by Equation (25) $\Pr\left[\mathcal{N}'_u = \emptyset\right] \to 0$. Equation (24) then becomes:

$$\begin{aligned}&\Pr\Big[\sum_{n\in\mathcal{N}_u\cup\{l\}} \widehat{y}_{nu} = 0 \mid \mathcal{N}'_u \neq \emptyset\Big] = \\ &\overset{(a)}{=} \prod_{n\in\mathcal{N}'_u} \left(1 - \frac{y^\dagger_{nu}}{x^\dagger_{ns_u}}\right) \cdot \left(1 - \left[\frac{y^\dagger_{lu} - \prod_{n\in\mathcal{N}_u}(1 - x^\dagger_{ns_u})}{1 - \prod_{n\in\mathcal{N}_u}(1 - x^\dagger_{ns_u})}\right]_+\right) \\ &\overset{(b)}{\leq} \prod_{n\in\mathcal{N}'_u} \left(1 - \frac{y^\dagger_{nu}}{x^\dagger_{ns_u}}\right) \end{aligned} \tag{26}$$

where (a) holds because the routing variables are rounded independently of one another and in the end one of them is picked (so the probability of not picking any of the variables is equal to the probability of all of them being rounded to zero). (b) holds because term $[.]_+$ is a probability value.

As BS density and service requirements increase, the set $\mathcal{N}'_u$ grows and eventually contains a BS n that is out of range of all the other users requesting the same service s_u. In this case, there is an optimal fractional solution with $y^\dagger_{nu}/x^\dagger_{ns_u} = 1$ since it would be wasteful for BS n to store a larger portion $x^\dagger_{ns_u}$ than the routed portion $y^\dagger_{nu}$, while Constraint (5) ensures that the stored portion is larger or equal to the routed one, so it should be equal, and the product in (26) goes to 0. □

We note that there can be also derived worst case results to upper bound the probability $\Pr\left[\sum_{n\in\mathcal{N}_u\cup\{l\}}\widehat{y}_{nu}=0\right]\leq 1/e\approx 0.3679$ in all cases and without the BS density and service requirement assumptions of Lemma 1. These worst case results can be derived by showing that the product function in (26) is Schur-concave and then applying majorization theory.

By construction, SPR^3 routes requests only to BSs that are nearby and have stored the respective service ($\mathcal{N}'_u$ set in Line 5) or to the cloud. Therefore, Constraints (4) and (5) are satisfied. Next, we study whether the remaining constraints in (6), (7), (8), and (9) are satisfied.

Lemma 2. *The solution returned by the SPR^3 algorithm satisfies in expectation the storage, computation, and bandwidth capacity constraints in (6), (7), (8), and (9).*

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

$$\mathbb{E}[\sum_{s\in\mathcal{S}}\widehat{x}_{ns}r_s]=\sum_{s\in\mathcal{S}}\Pr[\widehat{x}_{ns}=1]r_s=\sum_{s\in\mathcal{S}}x^{\dagger}_{ns}r_s=R_n \quad (27)$$

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 (6) 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:

$$\mathbb{E}[\sum_{u\in\mathcal{U}}\widehat{y}_{nu}c_{s_u}]=\sum_{u\in\mathcal{U}}\Pr[\widehat{y}_{nu}=1]c_{s_u}=\sum_{u\in\mathcal{U}}y^{\dagger}_{nu}c_{s_u}\leq C_n \quad (28)$$

where the second equation holds due to Equation (22). The inequality is by Constraint (7). Similar inequalities can be shown for the uplink/downlink bandwidth constraints:

$$\mathbb{E}[\sum_{u\in\mathcal{U}}\widehat{y}_{nu}b^{\uparrow}_{s_u}]=\sum_{u\in\mathcal{U}}\Pr[\widehat{y}_{nu}=1]b^{\uparrow}_{s_u}=\sum_{u\in\mathcal{U}}y^{\dagger}_{nu}b^{\uparrow}_{s_u}\leq B^{\uparrow}_n \quad (29)$$

$$\mathbb{E}[\sum_{u\in\mathcal{U}}\widehat{y}_{nu}b^{\downarrow}_{s_u}]=\sum_{u\in\mathcal{U}}\Pr[\widehat{y}_{nu}=1]b^{\downarrow}_{s_u}=\sum_{u\in\mathcal{U}}y^{\dagger}_{nu}b^{\downarrow}_{s_u}\leq B^{\downarrow}_n \quad (30)$$

where we have used Equations (8), (9) and (22). □

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:

$$\mathbb{E}[\sum_{u\in\mathcal{U}}\widehat{y}_{lu}]=\sum_{u\in\mathcal{U}}\Pr[\widehat{y}_{lu}=1]=\sum_{u\in\mathcal{U}}y^{\dagger}_{lu} \quad (31)$$

where the second equation holds due to Equation (21). □

The above lemmas have shown that the SPR^3 algorithm satisfies *in expectation* the capacity constraints and achieves the optimal objective value. However, *in practice*, the capacity 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 $n\in\mathcal{N}$ will not exceed its storage capacity by a factor larger than $\frac{3\ln(S)}{R_n}+4$ with high probability under the assumptions $R_n\geq\ln(S)$ and $S>N$.*

Proof. The proof uses the following Chernoff Bound [26]:

Given I independent variables $z_1,z_2,\ldots,z_I$ where for all $z_i\in[0,1]$ and $m=\mathbb{E}[\sum_{i=1}^{I}z_i]$, it holds that $\Pr[\sum_{i=1}^{I}z_i\geq(1+\epsilon)m]\leq\exp^{\frac{-\epsilon^2 m}{2+\epsilon}}$.

For a given BS $n\in\mathcal{N}$, the products $\widehat{x}_{ns}r_s$ for all $s\in\mathcal{S}$ are independent random variables with expected total value $\mathbb{E}[\sum_{s\in\mathcal{S}}\widehat{x}_{ns}r_s]=R_n$ (cf. Equation (27)). 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 [26] to show that for any $\epsilon>0$:

$$\Pr[\sum_{s\in\mathcal{S}}\widehat{x}_{ns}r_s\geq(1+\epsilon)R_n]\leq\exp^{\frac{-\epsilon^2 R_n}{2+\epsilon}} \quad (32)$$

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

$$\exp^{\frac{-\epsilon^2 R_n}{2+\epsilon}}\leq\frac{1}{S^3} \quad (33)$$

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:

$$\epsilon\geq\frac{3\ln(S)}{2R_n}+\sqrt{\frac{9\ln^2(S)}{4R_n^2}+\frac{6\ln(S)}{R_n}} \quad (34)$$

The above condition holds if we pick:

$$\epsilon=\frac{3\ln(S)}{R_n}+3 \quad (35)$$

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

$$\begin{aligned}&\Pr[\bigcup_{n\in\mathcal{N}}\sum_{s\in\mathcal{S}}\widehat{x}_{ns}r_s\geq(1+\epsilon)R_n]\\&\leq\sum_{n\in\mathcal{N}}\Pr[\sum_{s\in\mathcal{S}}\widehat{x}_{ns}r_s\geq(1+\epsilon)R_n]\\&\leq N\frac{1}{S^3}\leq\frac{1}{S^2}\end{aligned} \quad (36)$$

where the first inequality is due to the Union Bound theorem. The second inequality is due to Equation (33) 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 $n\in\mathcal{N}$ 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, under the assumptions $\lambda^{\dagger}\geq\ln(S)$ and $S>N$.*

Proof. The proof is similar to Theorem 1. For a given BS $n \in \mathcal{N}$, the variables $\widehat{y}_{nu}c_{s_u}$ for all $u \in \mathcal{U}$ 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. Equation (28)). Moreover, they can be normalized to take values within $[0,1]$. Therefore, we can apply the Chernoff Bound theorem:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)\sum_{u\in\mathcal{U}} y^{\dagger}_{nu}c_{s_u}] \leq \exp^{\frac{-\epsilon^2\sum_{u\in\mathcal{U}} y^{\dagger}_{nu}c_{s_u}}{2+\epsilon}} \quad (37)$$

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 (7)) 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:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)C_n] \leq \Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)\sum_{u\in\mathcal{U}} y^{\dagger}_{nu}c_{s_u}] \quad (38)$$

$$\exp^{\frac{-\epsilon^2\sum_{u\in\mathcal{U}} y^{\dagger}_{nu}c_{s_u}}{2+\epsilon}} \leq \exp^{\frac{-\epsilon^2\lambda^{\dagger}}{2+\epsilon}} \quad (39)$$

By combining Equations (37), (38), and (39), we obtain:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)C_n] \leq \exp^{\frac{-\epsilon^2\lambda^{\dagger}}{2+\epsilon}} \quad (40)$$

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

$$\begin{aligned}
&\Pr[\bigcup_{n\in\mathcal{N}}\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)C_n] \\
&\leq \sum_{n\in\mathcal{N}} \Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{nu}c_{s_u} \geq (1+\epsilon)C_n] \\
&\leq N\frac{1}{S^3} \leq \frac{1}{S^2}
\end{aligned} \quad (41)$$

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 $n \in \mathcal{N}$ 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, under the assumptions $\mu^{\dagger} \geq \ln(S)$ and $S > N$.*

Theorem 4. *The downlink bandwidth load of BS $n \in \mathcal{N}$ 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 under the assumptions $\nu^{\dagger} \geq \ln(S)$ and $S > N$.*

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 under the assumption $\xi^{\dagger} \geq \ln(S)$.*

Proof. The proof is similar to the previous theorems, yet the bound is tighter since we do not need to apply the Union Bound theorem. We begin by showing that:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{lu} \geq (1+\epsilon)\xi^{\dagger}] \leq \exp^{\frac{-\epsilon^2\xi^{\dagger}}{2+\epsilon}} \quad (42)$$

Since $\xi^{\dagger} \leq \widehat{\xi}$ where $\widehat{\xi}$ is the optimal integer solution value, it also holds that:

$$\Pr[\sum_{u\in\mathcal{U}} \widehat{y}_{lu} \geq (1+\epsilon)\widehat{\xi}] \leq \exp^{\frac{-\epsilon^2\xi^{\dagger}}{2+\epsilon}} \quad (43)$$

Next, we upper bound the right hand side of the above inequality by $1/S^2$. In order for this to be true, the ϵ value must satisfy the following condition:

$$\epsilon \geq \frac{\ln(S)}{\xi^{\dagger}} + \sqrt{\frac{\ln^2(S)}{\xi^{\dagger 2}_n} + \frac{4\ln(S)}{\xi^{\dagger}}} \quad (44)$$

The above condition holds if we pick:

$$\epsilon = \frac{2\ln(S)}{\xi^{\dagger}} + 2 \quad (45)$$

since, in practice, the number of requests will be more than the number of services ($\xi^{\dagger} \geq \ln(S)$). Thus, with high probability, performance will be at most $1+\epsilon = \frac{2\ln(S)}{\xi^{\dagger}}+3$ times worse than optimal. □

In many practical settings, the above factors are constant, i.e., the term that depends on the system parameters is small. For example, consider a system with thousands of users generating requests for services in a library of size $S = 1,000$. Each BS can process up to a thousand requests ($C_n = 1,000$) and the minimum computation capacity utilization is 40% ($\lambda^{\dagger} = 400$). Then, the computation capacity violation factor becomes $\frac{3\ln(1000)}{400}+4 \approx 4.05$.

The factors in Theorems 1-5 upper bound with probability that goes to 0 at a quadratic rate the violation of capacity constraints and the performance gap from optimal. While we showed that each of the five factors holds *individually*, we can also show that the five factors hold *jointly*, essentially binding the factors together. This can be achieved by using again the Union Bound Theorem where the probability that *any* of the five factors in Theorems 1-5 does not hold goes to zero at a quadratic rate $5/S^2$ which is the sum of the right-hand sides of Equations (36), (41) and analog equations in Theorems 3-5.

We need to emphasize that our analysis exploits the *large scale* of the MEC system (large number of base stations and services) to derive the high probability bounds of constraint violation and performance gap caused by the rounding process of the SPR^3 algorithm. Another promising direction is to

repeat the rounding process of SPR^3 multiple times and exploit this *redundancy* to prove tighter bounds. For example, the pioneer work in [18] performs multiple rounding tries until a multi-criteria solution is found, while our previous work in [19] bounds the probability of constraint violation as a function of the rounding tries. Such repeated rounding process could be also added to our SPR^3 following similar arguments as in [18] and [19]. The improved bounds, however, would be at the cost of increased computation time of the algorithm.

V. Extension and Practical Cases

In this section, we discuss how to handle changes in the user demand. In addition, we describe how to make the solution of the SPR^3 algorithm satisfy the constraints, thereby making the algorithm more practical.

A. Handling user demand changes

The service placement and request routing decisions are taken for a certain time period during which the demand is fixed and predicted. The demand, however, may change over time, e.g., after a few hours or even at a faster timescale depending on the scenario. The MEC operator will have to repeatedly predict the new demand for the next time period and *adapt* the service placement and request routing decisions accordingly. For example, the MEC operator should replace services that are no longer popular with other services that recently gained popularity.

The adaptation of the service placement is not without cost. In fact, replacing previously placed services with new ones would require the BSs to download non-trivial amounts of data from the cloud through their backhaul links. This operation creates overheads which, depending on the timescale, can be significant and therefore should be avoided.

The SPR^3 algorithm can be extended to become aware of the service placement adaptation costs. To this end, we add a new constraint into the JSPRR problem. This constraint upper bounds by a constant D the total amount of data associated with the replaced services:

$$\sum_{n\in\mathcal{N}}\sum_{s\in\mathcal{S}} x_{ns}(1-x^p_{ns})r_s \leq D \tag{46}$$

where x^p_{ns} is the placement solution in the previous time period. Here, placing a service s at BS n ($x_{ns}=1$) adds r_s to the adaptation cost if and only if that service was not placed in the previous time period ($x^p_{ns}=0$).

We note that all the presented lemmas and theorems still hold as they do not depend on the presence of constraint (46). What remains to analyze is how likely is for the rounded solution $\widehat{\boldsymbol{x}}$ returned by the algorithm to violate constraint (46). This is described in the following theorem.

Theorem 6. *The total amount of data associated with service placement adaptation will not exceed the upper bound D by more than a factor of $\frac{2\ln(S)}{D}+3$ with high probability.*

Proof. The proof is similar to the previous theorems. The Chernoff Bound is applied for the sum of random variables $\{x_{ns}(1-x^p_{ns})r_s\}$, the expected total value of which is D. □

B. Constructing a feasible solution

As the SPR^3 algorithm may violate the storage capacities of the BSs by a factor of $3\ln(S)/R_n+4$, the MEC operator may not be able to store all the services required to ensure the performance guarantee of the algorithm. Similarly, the service placement may violate the limit of allowable adaptations D, while the request routing may overwhelm the computation and bandwidth capacities or leave some users unserved. To respond to such cases, the operator needs to convert the multi-criteria solution into a *feasible* solution, i.e., a solution that satisfies constraints (3), (6)-(9) and (46).

To obtain such a solution, we start with the service placement $\widehat{\boldsymbol{x}}$ outputted by the SPR^3 algorithm. Then, we iteratively perform the removal of a service from a BS that yields the minimum cloud load increment. When a service is removed from a BS, the user requests for that service previously routed to that BS are now re-directed to other nearby BSs with available bandwidth and computation and the requested service stored (if any), or otherwise to the cloud. The procedure ends when constraints (6) and (46) are satisfied. To satisfy the remaining constraints we perform one more step. That is, we iteratively re-direct a user request that is unserved or served by an overloaded BS towards another BS with available resources (if any) or to the cloud, until all the requests are served without any overloaded BSs. The re-directions of requests happen by ranking the users and BSs based on their indices and examining each user-BS pair one-by-one following that order, while ensuring that none of the constraints is violated because of each re-direction.

VI. Evaluation Results

In this section, we carry out evaluations to show the performance of the proposed SPR^3 algorithm after we convert its solution into a feasible one (Section V.B). We consider a similar setup as in the previous work [9], depicted in Figure 3. Here, $N=9$ base stations (BSs) are regularly deployed on a grid network inside a $500\text{m}\times500\text{m}$ 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, which is a common assumption for several types of services such as video streaming. 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 $B^{\uparrow}_n=100$ ($B^{\downarrow}_n=250$) Mbps. Yet, all these values are varied during the evaluations.

We set the resource requirements r_s, c_s, $b^{\uparrow}_s$ and $b^{\downarrow}_s$ 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), listed in Table I. **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

TABLE I: Resource requirements of different types of services.

Service s	Uplink $b_s^{\uparrow}$ (Mbps)	Storage r_s (GB)	Computation c_s (GHz)	Downlink $b_s^{\downarrow}$ (Mbps)
Video Streaming (VS)	-	[1,10]	-	[1,25]
Face Recognition (FR)	[1,8]	[2,10]	[0.375,3]	-
Gzip	[1,8]	0.02	[0.04,0.32]	[0.25,2]
Augmented Reality (AR)	[1,8]	[2,20]	[0.375,3]	[0.25,2]

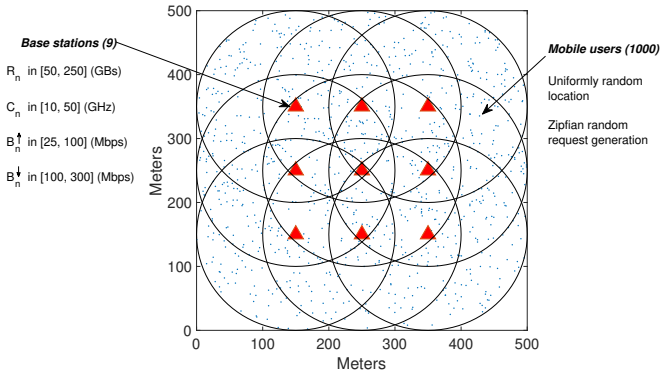


Fig. 3: Evaluation setup.

this particular service. **Face recognition** consumes notable uplink bandwidth for video frame uploading (1Mbps to 8Mbps) which depends on the resolution of the camera (SD or HD) and the use case of interest (e.g., security and surveillance or access control). 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. The above values are inline with the real service specifications in [27]. **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 assuming 330 cycles per byte (or about 40 cycles per bit) of the uploaded data [28] 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 similar to the FR service while the required storage can be more than 10 GBs [29].

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 [15]:* 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 constraints.

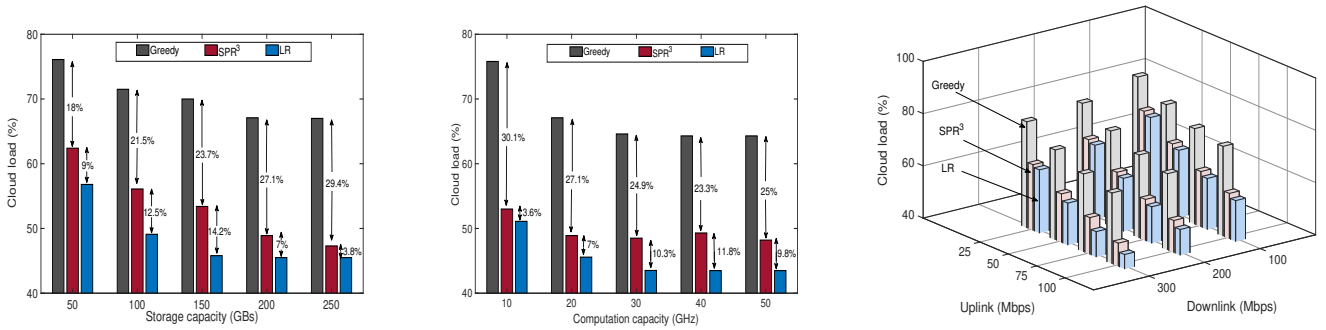
On one hand, LR can be used as a benchmark to gauge the performance gap of our algorithm from optimal. On the other hand, it is well-known that Greedy achieves near-optimal performance for the traditional data placement (or caching) problem, leveraging its submodular property [15]. Therefore, a natural question to ask is whether the efficiency of Greedy is maintained or novel algorithms are needed when the placement of services with multidimensional resource requirements is considered. We remark that our evaluation code is publicly available online in [40].

We first explore the impact of storage capacity R_n $\forall n$ on the load of the centralized cloud. In Figure 4a, 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% 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 4b. 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 4c 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.

We take a closer look into the utilization of BS resources when the SPR^3 and Greedy algorithms are used. The four subplots in Figure 5 show the resource utilization for each of the four resource types. We observe that both algorithms utilize most of the available storage and computation resources (90% or more for most BSs). Interestingly, Greedy utilizes slightly more of these two types of resources. Yet, SPR^3 manages to utilize significantly more bandwidth for almost all BSs. Specifically, it utilizes 95.7% of BS uplink bandwidth on average as opposed to 85.7% of the Greedy. The difference is more pronounced for the downlink bandwidth. Such effective balancing of load and utilization of bandwidth resources eventually leads to superior performance.

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. To shed light on this issue, Figure 6 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



(a) Impact of storage capacity. (b) Impact of computation capacity. (c) Impact of bandwidth capacities.

Fig. 4: Cloud load for different (a) storage, (b) computation, and (c) uplink/downlink bandwidth capacities of BSs.

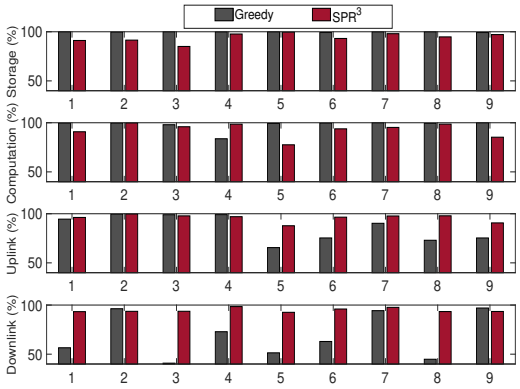


Fig. 5: Resource utilization per type and per BS.

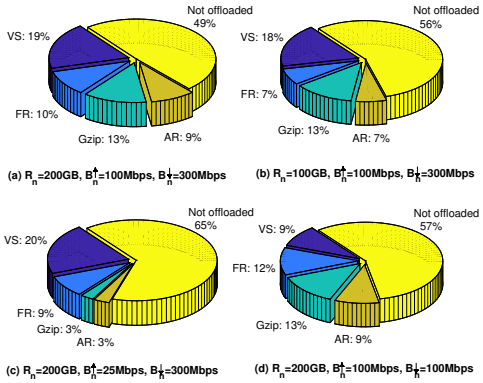


Fig. 6: Distribution of requests across services for different resource capacities: (a) default values, (b) reduced storage, (c) reduced uplink and (d) reduced downlink.

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%. This was expected since video streaming has the highest downlink rate requirements (up to 25Mbps).

The above results revealed that our SPR^3 algorithm tends to offload fewer requests for resource-demanding services (e.g., AR), depending on which are the bottleneck resources, so as to maximize the aggregate of offloaded requests for all the services. This is desirable when all the services are fairly sensitive to latency. Yet, in some cases, certain services may be more latency-sensitive than others and hence it is more critical to prioritize their offloading. Our algorithm can be easily extended to handle such cases. Specifically, we can update the objective function in (10) by multiplying the variable y_{lu} for each user u requesting the critical service with a weight value. The larger the value of this weight is, the higher priority will be given by the algorithm to offload the requests for the critical service. Consider for example the case in Figure 7 where the weight for the AR service is varied from 1 to 10. While about 41% of the AR requests are offloaded for weight equal to 1, this percentage notably increases up to 62% as the weight increases. At the same time, the aggregate of offloaded requests decreases in a mediocre way, as some of the requests for the other services are "sacrificed" to favor the AR service.

Another interesting question is how much the resource capacities are violated by algorithm before its solution is con-

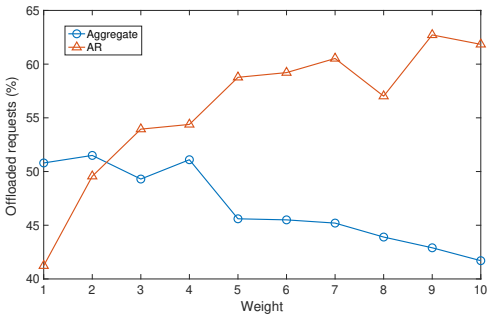


Fig. 7: Impact of weight value in the objective function on the aggregate and AR offloaded requests.

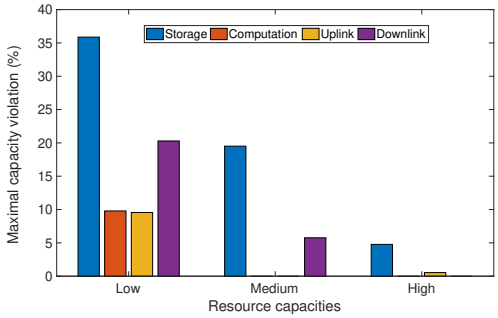


Fig. 8: Violation of resource capacities in three scenarios.

verted into a feasible one. Figure 8 shows the maximal (over all BSs) capacity violation for each type of resource. Here, three scenarios are investigated differing in the availability of storage, computation, uplink and downlink resources, namely low (25GB, 5GHz, 25Mbps, 100Mbps), medium (50GB, 10GHz, 50Mbps, 200MBps), and high (250GB, 50GHz, 100Mbps, 300Mbps). Overall, the capacity violations are much smaller than the worst case conditions in theorems 1-4 indicate. In the worst case (low scenario), the storage capacity is violated by about 35%. Yet, the violation factors become negligible as the capacities increase.

So far, we have assumed that the demands of the various services can be estimated accurately. In practice, such estimation is unlikely to have perfect accuracy and some estimation error will exist. For example, estimators that rely on historical records to predict future demands cannot estimate the demands for new services that recently entered the market and hence they do not have any records. We denote the number of these new services by L and refer to it as the "aging factor" since these new services represent how "aged" the records of the estimator are. We then perform additional evaluations where the actual service popularity follows a different distribution than the estimated one (Zipf distribution considered so far). Specifically, we set the actual popularity distribution to be of size $S + L$ where the L new services are augmented to the end of the estimated popularity vector. Then, the actual service requested by a user (s_u) is randomly set to either the estimated

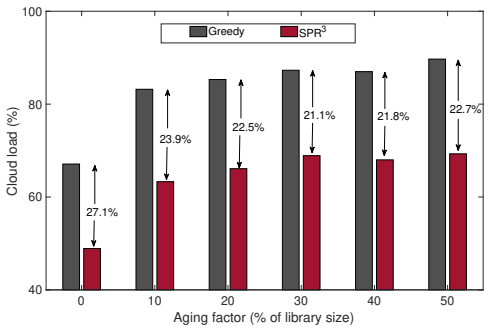


Fig. 9: Impact of inaccurate estimation of user demand.

TABLE II: Running times for different algorithms.

Algorithm	Running time (seconds)			
	U=500	U=1000	U=1500	U=2000
LR	0.6	2.1	6.8	27.6
SPR^3	0.7	2.2	7.0	27.7
Greedy	19.8	23.6	54.0	64.8

service or one of the services in the L subsequent positions of the new popularity vector. Figure 9 depicts the impact of the aging factor L on the cloud load returned by SPR^3 and Greedy. As expected, aging affects the performance of both algorithms. However, SPR^3 performs consistently better than Greedy, which shows the robustness of our method.

We finally highlight the running times of the presented algorithms, summarized in Table II. These running times are based on a Matlab implementation run on a MacBook Pro laptop with 2.3 GHz Intel Core i5 processor. The proposed SPR^3 algorithm requires only slightly more time than LR to perform the rounding process regardless of the number of users U. Greedy requires significantly more time to return a solution and is 2.3 to 28 times slower than the proposed algorithm.

VII. Related Works

Most of the existing related works considered only some of the three types of resources (computation, communication, and storage). These works can be grouped into five categories: (i) computation and communication, (ii) caching, (iii) caching and communication, (iv) caching and computation, and (v) caching, computation, and communication. We describe these works in the sequel and list them in Table III.

A. Computation & Communication

The most common approach in the literature is to treat services as virtual network slices that consume two types of resources, computation and communication. In this context, the works of [4] and [5] proposed methods to decide which service requests to *offload* to the edge cloud servers or to execute locally at the mobile devices aiming to minimize delay and computation overhead. Another line of research focused on the problem of *placing middleboxes* that are able to host services. The question of placing a minimum number of middleboxes together with the routing of traffic flows through them subject

TABLE III: RELATED WORKS ON COMPUTATION, COMMUNICATION & CACHING.

Ref.	Computing	Communication	Caching	Objective	Technique	Solution
[4]	✓	✓	✗	Min delay	Convex optimization	Heuristic
[5]	✓	✓	✗	Min computation overhead	Greedy	Heuristic
[21], [22]	✓	✓	✗	Min # middleboxes	Submodularity	$\log(\min(N, B))$-approx.
[23]	✓	✓	✗	Min # middleboxes	Randomized rounding	Multi-criteria-approx.
[16]	✓	✓	✗	Min traffic cost	Markov approximation	Heuristic
[17]	✓	✓	✗	Min bandwidth	Column generation	ϵ-optimal
[18]	✓	✓	✗	Max profit	Randomized rounding	Multi-criteria approx.
[20]	✓	✓	✗	Min resource cost	Linear programming	Optimal
[13]	✗	✗	✓	Min delay	Relaxation & rounding	10-approx.
[14]	✗	✗	✓	Min delay	Greedy & swapping	$1/2$-approx.
[15]	✗	✗	✓	Min delay	Submodularity	$e/(e-1)$-approx.
[30]	✗	✓	✓	Max cache hits	Lagrangian relaxation	Heuristic
[31]	✗	✓	✓	Max cache hits	Facility location	O(S)-approx.
[32]	✗	✓	✓	Min delay	Submodularity	$e/(e-1)$-approx.
[33]	✗	✓	✓	Min transmit power	Learning & clustering	Heuristic
[34]	✗	✓	✓	Min brown energy	Sequential fixing	Heuristic
[35]	✓	✗	✓	Max quality	Stochastic knapsack	Heuristic
[36]	✓	✗	✓	Min delay	Conditional gradient	Heuristic
[37]	✓	✗	✓	Max revenue	ADMM	Heuristic
[10]	✓	✓	✓	Max cache hits	Submodularity	Heuristic
[38]	✓	✓	✓	Min cost & delay	BSUM	Heuristic
[39]	✓	✓	✓	Min energy & bandwidth	ADMM	Heuristic
This work	✓	✓	✓	Max cache hits	Randomized rounding	Multi-criteria-approx.

to computation constraints was formulated as a submodular optimization problem for which efficient approximation algorithms are known [21], [22]. Another approximation algorithm for the same problem was provided in [23] using randomized rounding techniques. The problem of placing virtual machines (or functions) and routing traffic in a network was studied in [16] and a Markov approximation was given. An extension of this work for flows that require to traverse *chains* of functions in specific order was provided in [17]. The authors applied the column generation method to approximate the solution with the minimum bandwidth cost. This problem can be also casted as a virtual network embedding (VNE) problem for which multi-criteria approximation algorithms are known [18]. Extensions of the VNE approach to handle multicast routing were recently given in [19]. The fractional analog of integral routing and function placement was studied in [20], which led to tractable linear programming formulations. However, the above works focused on the computation and communication resources, neglecting that, for many services, non-trivial amounts of data need to be stored at the servers.

B. Caching

Another related problem is the data placement or caching problem, which asks to place popular contents into caches distributed throughout a network, given some predicted distribution of content demand. Approximation algorithms have been developed by applying linear relaxation and rounding in [13], greedy and swapping methods in [14], and submodular optimization in [15]. However, the caching problem only considers storage ignoring the other types of resources.

C. Caching & Communication

Recently, the caching problem was extended to account for the communication between caches and users requesting the contents, which can be the bottleneck resource. Specifically, [30] formulated the *joint content caching and request routing problem* under link bandwidth constraints so that network congestion is avoided and the volume of served requests by caches is maximized. A Langragian relaxation method tailored to hierarchical cache topologies was presented. The same problem was studied in [31] for a two-tier caching network with caches installed in macro-cells and small-cells for which a facility location inspired approximation was proposed. The submodularity property was used again in [32] for a similar network setup where the greedy algorithm was used. For a 2-tier caching network formed by drones and infrastructure nodes, learning and clustering techniques were applied in [33]. For cache-nodes that operate using renewable energy, a sequential-fixing algorithm was proposed in [34].

D. Caching & Computation

Traditionally, caching and computation were treated as two separate resource allocation problems. Joint caching and computation frameworks were recently introduced in MEC networks for satisfying user requests for videos at different bitrates. Specifically, a lower bitrate variant can be obtained from a higher bitrate variant via transrating or transcoding, a process that consumes computation resources. [35] and [36] optimized the caching of videos and allocation of computation resources so as to improve video quality and delay. Another joint caching and computation problem was considered in [37] for offloading mining tasks and caching cryptographic hashes of blocks in a blockchain network, for which an

ADMM algorithm was proposed. However, these works did not consider the bandwidth allocation problem.

E. Caching, Communication & Computation

Only a few works have considered all the three types of resources. The work in [10] studied joint service placement and request scheduling in edge cloud systems. However, it assumed that the coverage regions of the base stations are non-overlapping and therefore each user can associate with only one base station. The submodularity property was shown for the special cases of unit-sized storage capacities and non-congestible computation capacities. A block successive upper bound minimization (BSUM) method was proposed in [38] for allocating triplets of resources. However, no hard bandwidth constraints were considered for the links. This assumption was relaxed in [39] which applied the ADMM method. However, these works did not provide optimal or approximate solutions for the joint service placement and request routing problem.

VIII. Conclusion

In this paper, we studied service placement and request routing in MEC-enabled multi-cell networks with 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 (e.g., AR). 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. Interesting directions for future work include studying the coordination between BSs through backhaul links as well as the generalization of our model to services with multiple (chained) functions.

References

[1] K. Poularakis, J. Llorca, A.M. Tulino, I. Taylor, L. Tassiulas, "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks", *IEEE Infocom*, 2019.

[2] 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.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, "Mobile Edge Computing - A Key Technology Towards 5G", *ETSI White Paper*, 2015.

[4] 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.

[5] 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.

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

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

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

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

[10] 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.

[11] 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 Comm., vol. 25, no. 3, pp. 21-27*, 2018.

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

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

[14] S. Borst, V. Gupta, A. Walid, "Distributed Caching Algorithms for Content Distribution Networks", *IEEE Infocom*, 2010.

[15] 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.

[16] J.W. Jiang, T. Lan, S. Ha, M. Chen, Mung Chiang, "Joint VM Placement and Routing for Data Center Traffic Engineering", *IEEE Infocom*, 2012.

[17] N. Huin, B. Jaumard, and F. Giroire, "Optimal Network Service Chain Provisioning", *IEEE/ACM Trans. Networking, vol. 26, no. 3*, 2018.

[18] M. Rost, S. Schmid, "Virtual Network Embedding Approximations: Leveraging Randomized Rounding", *IEEE/ACM Transactions on Networking, vol. 27, no. 5, pp. 2071-2084*, 2019.

[19] M. Michael, J. Llorca, A.M. Tulino, "Approximation Algorithms for the Optimal Distribution of Real-Time Stream-Processing Services", *IEEE ICC*, 2019.

[20] H. Feng, J. Llorca, A.M. Tulino, D. Raz, A.F. Molisch, "Approximation Algorithms for the NFV Service Distribution Problem", *Infocom*, 2017.

[21] T. Lukovszki, M. Rost, S. Schmid, "It's a Match! Near-Optimal and Incremental Middlebox Deployment", *ACM SIGCOMM Comput. Commun. Rev., vol. 46, no. 1, pp. 30-36*, 2016.

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

[23] M. Charikar, Y. Naamad, J. Rexford, X.K. Zou, "Multi-Commodity Flow with In-Network Processing", *ALGOCLOUD*, 2018.

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

[25] Y.T. Lee, A. Sidford, "Efficient Inverse Maintenance and Faster Algorithms for Linear Programming", *FOCS*, 2015.

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

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

[28] 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.

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

[30] J. Dai, Z. Hu, B. Li, J. Liu, B. Li, "Collaborative Hierarchical Caching with Dynamic Request Routing for Massive Content Distribution", in *Proc. of IEEE Infocom*, 2012.

[31] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation Algorithms for Mobile Data Caching in Small Cell Networks", *IEEE Trans. Comm., vol. 62, no. 10, pp. 3665-3677*, 2014.

[32] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, R. Sitaraman, "On the Complexity of Optimal Request Routing and Content Caching in Heterogeneous Cache Networks", *IEEE/ACM Tran. Netw., vol. 25, no. 3, pp. 1635-1648*, 2017.

[33] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the Sky: Proactive Deployment of Cache-enabled Unmanned Aerial Vehicles for Optimized Quality-of-Experience", *IEEE J. Sel. Areas Commun., vol. 35, no. 5, pp. 1046–1061*, 2017.

[34] A. Khreishah, H. B.-Salameh, I. Khalil, A. Gharaibeh, "Renewable energy-aware joint caching and routing for green communication networks", *IEEE Syst. J.*, vol. 12, no. 1, pp. 768–777, Mar. 2018.

[35] H. A. Pedersen, S. Dey,"Enhancing Mobile Video Capacity and Quality Using Rate Adaptation, RAN Caching and Processing," *IEEE/ACM Trans. on Networking, vol. 24, no. 2, pp. 996-1010*, 2016.

[36] T.X. Tran, D. Pompili, "Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks", *IEEE Trans. Mobile Comp.*, 2018.

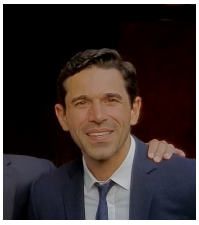
[37] M. Liu, F.R. Yu, Y. Teng, V.C.M. Leung, M. Song, "Computation Offloading and Content Caching in Wireless Blockchain Networks with Mobile Edge Comp.", *IEEE Trans. on Vehicular Tech.*, 2018.

[38] A. Ndikumana, N. H. Tran, T.M. Ho, Z. Han, W. Saad, D. Niyato, C.S. Hong, "Joint Communication, Computation, Caching, and Control in Big Data Multi-access Edge Computing", *IEEE Trans. Mobile Comp.*, 2019.

[39] Q. Chen, F.R. Yu, T. Huang, R. Xie, J. Liu, Y. Liu, "Joint Resource Allocation for Software-Defined Networking, Caching, and Computing", *IEEE Transactions on Networking, vol. 26, no. 1, pp. 274-287*, 2018.

[40] https://www.dropbox.com/s/gutw8milc4ttl00/main.m?dl=0

Konstantinos Poularakis obtained the Diploma, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Thessaly, Greece, in 2011, 2013 and 2016 respectively. In 2014, he was a Research Intern with Technicolor Research, Paris. Currently, he is a Research Scientist at Yale University. His research interests lie in the area of network optimization with emphasis on emerging architectures such as software defined networks and mobile edge computing and caching networks. He was the recipient of several awards and scholarships during his studies, from sources including the Greek State Scholarships foundation (2011), the Center for Research and Technology Hellas (2012), the Alexander S. Onassis Public Benefit Foundation (2013) and the Bodossaki Foundation (2016). He also received the Best Paper Award at the IEEE Infocom 2017 and IEEE ICC 2019.



Jaime Llorca (S '03 – M '09) received the B.Sc. degree in Electrical Engineering from Universidad Politecnica de Catalunya, Barcelona, Spain, in 2001, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from University of Maryland, College Park, MD, in 2003 and 2008, respectively. He held a post-doctoral position with the Center for Networking of Infrastructure Sensors, College Park, MD, from 2008 to 2010, and a Senior Research Scientist position with Bell Labs, New Jersey, from 2010 to 2019. He is currently a Research Professor with the Electrical and Computer Engineering Department, New York University Tandon School of Engineering, Brooklyn, NY. His research interests are in the field of network algorithms, network optimization, distributed control, and network information theory, with applications to next generation communication networks, distributed cloud networking, and content distribution. He currently serves as an Associate Editor for the IEEE Transactions on Networking. He is a recipient of the 2007 Best Paper Award at the IEEE International Conference on Sensors, Sensor Networks and Information Processing, the 2016 Best Paper Award at the IEEE International Conference on Communications, and the 2015 Jimmy H.C. Lin Award for Innovation.

Antonia M. Tulino (F'13) received the Ph.D. degree in electrical engineering from Seconda Universita degli Studi di Napoli, Italy, in 1999. She held research positions with the Center for Wireless Communications, Princeton University, Oulu, Finland, and also with the Universita degli Studi del Sannio, Benevento, Italy. From 2002 to 2016, she has been an Associate Professor with the Universita' degli Studi di Napoli Federico II, where she is, now, Full Professor since 2017. Since 2009, she has been collaborating with Nokia Bell Labs. Starting from October 2019, she is, also, Research Professor at the at Dep. of Electrical and Computer Engineering NYU Tandon School of Engineering, NY, 11201, USA. Her research interests lay in the area of communication systems approached with the complementary tools provided by signal processing, information theory, and random matrix theory. From 2011 to 2013, she has been a member of the Editorial Board of the IEEE Transactions on Information Theory and in 2013, she was elevated to IEEE Fellow. From 2019, she is the chair of the Information Theory society Fellows Committee. She has received several paper awards and among the others the 2009 Stephen O. Rice Prize in the Field of Communications Theory for the best paper published in the IEEE Transactions on Communications in 2008. She was a recipient of the UC3M-Santander Chair of Excellence from 2018 to 2019. She has been a principal investigator of several research projects sponsored by the European Union and the Italian National Council, and was selected by the National Academy of Engineering for the Frontiers of Engineering program in 2013.

Ian Taylor is a Professor at the University of Notre Dame and a Reader at Cardiff University. He has a degree in Computing Science, a Ph.D. studying neural networks applied to musical pitch and he designed/implemented the data acquisition system and Triana workflow system for the GEO600 gravitational wave project. He now specializes in Blockchain, open data access, Web dashboards/APIs and workflows. Ian has published over 180 papers (h-index 43), 3 books and has won the Naval Research Lab best paper award in 2010, 2011 and 2015. Ian acts as general chair for the WORKS Workflow workshop yearly at Supercomputing.

Leandros Tassiulas (S'89, M'91, SM/05 F/07) is the John C. Malone Professor of Electrical Engineering at Yale University. His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models and algorithms of complex networks, architectures and protocols of wireless systems, sensor networks, novel internet architectures and experimental platforms for network research. His most notable contributions include the max-weight scheduling algorithm and the back-pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems. Dr. Tassiulas is a Fellow of IEEE (2007). His research has been recognized by several awards including the IEEE Koji Kobayashi computer and communications award, the inaugural INFOCOM 2007 Achievement Award for fundamental contributions to resource allocation in communication networks, the INFOCOM 1994 and 2017 best paper awards, a National Science Foundation (NSF) Research Initiation Award (1992), an NSF CAREER Award (1995), an Office of Naval Research Young Investigator Award (1997) and a Bodossaki Foundation award (1999). He holds a Ph.D. in Electrical Engineering from the University of Maryland, College Park (1991). He has held faculty positions at Polytechnic University, New York, University of Maryland, College Park, and University of Thessaly, Greece.