Joint Network Slicing and Mobile Edge Computing in 5G Networks

Bin Xiang
Politecnico di Milano
bin.xiang@polimi.it

Jocelyne Elias
Paris Descartes University
jocelyne.elias@parisdescartes.fr

Fabio Martignon
University of Bergamo
fabio.martignon@unibg.it

Elisabetta Di Nitto
Politecnico di Milano
elisabetta.dinitto@polimi.it

***Abstract*—Mobile traffic generated by a variety of services is rapidly increasing in volume. Both network and computation resources in a single edge network are therefore often too limited to provide the desired Quality of Service (QoS) to mobile users. In this paper, we propose a mathematical model, called JSNC, to perform an efficient joint slicing of mobile network and edge computation resources. JSNC aims at minimizing the total latency of transmitting, outsourcing and processing user traffic, under the constraint of user tolerable latency for multiple classes of traffic. The constraints of network, link and server capacities are considered as well. The optimization model results in a mixed-integer nonlinear programming (MINLP) problem. To tackle it efficiently, we perform an equivalent reformulation, and based on that, we further propose two effective heuristics: Sequential Fixing (SF), which can achieve near-optimal solutions, and a greedy approach which obtains suboptimal results with respect to SF. Both of them can solve the optimization problem in a very short computing time. We evaluate the performance of the proposed model and heuristics, showing the impact of all the considered parameters (viz. different types of traffic, tolerable latency, network topology and bandwidth, computation and link capacity) on the optimal and approximate solutions. Numerical results demonstrate that JSNC and the heuristics can provide efficient resource allocation solutions.**

***Index Terms*—Network slicing, edge computing, joint allocation, hierarchical network structure.**

I. Introduction

5G networks aim to meet different users' Quality of Service (QoS) requirements in several different application scenarios and use cases. Among the others, *latency* is certainly one of the key QoS requirements that mobile operators have to deal with. In fact, the classification devised by the International Telecommunications Union-Radio communication Sector (ITU-R), shows that mission-critical services depend on strong latency constraints. For example, in some use cases (e.g., autonomous driving), the tolerable latency is expected to reach less than 1 ms [1].

To address such constraints various ingredients are emerging. First of all, through *Network Slicing*, the physical network infrastructure can be split into several isolated logical networks, each dedicated to applications with specific latency requirements, thus enabling an efficient and dynamic use of network resources [2].

Second, *Multi-access Edge Computing (MEC)* provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network and in close proximity to mobile subscribers [3]. By combining MEC and network slicing, we have the possibility to address strong latency requirements of critical services still ensuring highly efficient network operation and service delivery, and offering an improved user experience.

In this line, we study the case of a complex network organized in multiple edge clouds, each of which connected to the Radio Access Network of a certain location. All such edge clouds are connected through various topologies, typically organized in some kind of hierarchy. This way, each edge cloud can serve end user traffic by relying not only on its own resources, but also offloading some traffic to its neighbors when needed. We specifically consider multiple classes of traffic and corresponding requirements, including voice, video, web, among others. For every class of incoming traffic, the corresponding edge cloud decides whether to serve it or to offload it to some other edge cloud node. This decision depends on the QoS requirements associated to the specific class of traffic and on the current status of the edge cloud.

Our main objective is to ensure that the infrastructure is able to serve all possible types of traffic within the boundaries of their QoS requirements and of the available resources. Therefore, our resource allocation approach, which we call JSNC-*Joint Slicing Network and Computation*, aims at minimizing the total traffic latency of transmitting, outsourcing and processing user traffic, under a constraint of user tolerable latency for each class of traffic. This optimization turns out to be a mixed-integer nonlinear programming (MINLP) one, which is an $\mathcal{NP}$-hard problem [4]. To tackle this challenge, we transform it into an equivalent mixed-integer quadratically constrained programming (MIQCP) problem, which can be solved more efficiently by the Branch and Bound method. Based on this reformulation, we further propose an effective heuristic, named Sequential Fixing, that permits obtaining near-optimal solutions in a very short computing time, even for the large-scale scenarios. Furthermore, we propose a simple heuristic, based on a greedy approach, that obtains slightly suboptimal solutions with respect to the sequential fixing approach, and still very fast. Finally, we systematically analyze and discuss with a thorough numerical evaluation the impact of all considered parameters (viz. different types of traffic, tolerable latency, network topology and bandwidth, computation and link capacity) on the optimal and approximate solutions obtained from our proposed model and heuristics. Numerical results demonstrate that our proposed model and heuristics can

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

provide very efficient resource allocation solution for multiple edge networks.

The remainder of this paper is organized as follows. Section II discusses related work. Section III introduces the network system architecture we consider. Section IV illustrates the proposed mathematical model and heuristics. Section V discusses numerical results in a set of typical network topologies and scenarios. Finally, Section VI concludes the paper.

II. Related Work

Workload offloading, that is, the possibility to delegate to other nodes the management of a certain workload, is considered in other approaches along with resource management (see [5]–[10]). More specifically, in [5], the authors propose to exploit mobile edge computing for jointly optimizing computation offloading, resource allocation and content caching in cellular networks. In [6] the on-line joint radio and computational resource allocation problem is studied with the goal to minimize the long-term average weighted sum of power consumption of the mobile devices and the MEC server. In [7], the authors investigate a MEC system with energy harvesting devices, and propose to jointly consider the offloading decision and allocation of the CPU-cycle frequencies and transmit power. The work in [8] aims at optimizing energy efficiency by a joint design of virtual computational resources, transmit beamforming, Remote Radio Head (RRH) selection, and RRH-user association. In [9], a framework is proposed to perform workload offloading decisions for mobile users and optimize the resource usage of mobile edge and the cloud by considering the tradeoff between system delay and cost. Finally, the authors of [10] study the task admission and resource allocation for MEC, while minimizing the energy consumption under the latency constraints of devices.

While the papers mentioned above mainly consider a single MEC system, in [11], the authors study task offloading and wireless resource allocation in an environment with multiple MEC servers. The work in [12] investigates collaborative computation offloading in a multi-access edge computing environment, while [13] studies the computing task offloading in the context of small-cell base-station (SBS) networks, where each SBS is equipped with an MEC server. However, they only consider flat MEC nodes.

Differently from the above discussed related work, in this paper we address the resource orchestration/allocation problem in 5G networks leveraging both network slicing and MEC. We explicitly model the delay due to computational resources and workloads in a multiple MEC system encountered by the mobile traffic at the access part of the mobile network (i.e., within network slices) as well as in the MEC. The scenario we model and analyze is more realistic compared to the others we have mentioned and, therefore, it results in an allocation approach that is more precise than the others.

III. Network System Architecture

Figure 1 illustrates our reference network system architecture. We consider a multiple edge network composed of

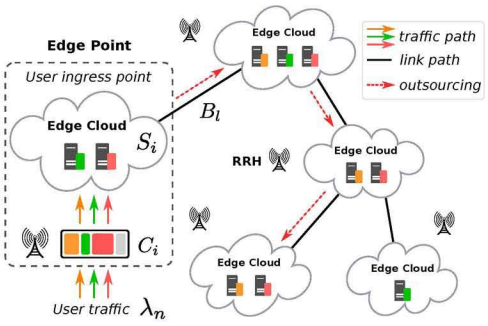

Fig. 1: Network System architecture.

edge points. Each edge point i includes both an *edge mobile network* of a specific capacity C_i and a co-located *edge cloud* of computation capacity S_i. We assume that different users' traffics are aggregated according to their *service types*. Different slices of the mobile network capacity C_i and edge cloud computation capacity S_i can be allocated to serve different types of user traffic based on the Service Level Agreements (SLAs) or user requirements, the colors in the figure represent such slicing. Each link l between different edge points has a fixed bandwidth, denoted by B_l.

The JSNC model is defined from the perspective of a *user ingress point* (the one in the dash square box of the figure) that is receiving user traffic and, when needed, offloads it to the other interconnected edge clouds. The same modeling approach can be applied to any of the edge points in the figure when they receive traffic from their connected edge network.

From the ingress edge network, all types of user traffic can be split and processed on all edge clouds. The red dashed arrows shown in the figure represent a possible outsourcing path of some traffic pieces.

In our scenario, we assume that the available network capacity and computation capacity of each edge network are known, for example through broadcast messages exchanged in the network. The bandwidth of each link in the network topology can be also estimated through periodic measurements. However, in real scenarios, these network parameters would change dynamically. We cope with this problem by considering a time-slotted system, where time is divided into equal-length slots (short periods where network parameters can be considered as fixed), and we study the optimal allocation of network and computation resources inside such slots to minimize the total latency of all traffic under latency constraints for each type of traffic. The same optimization process can be repeated for consecutive slots since, as shown in Section V, the computing time of our proposed heuristics permits doing so.

IV. Problem Formulation and Heuristics

In this section we first formulate our Joint Slicing of mobile Network and edge Computation resources (JSNC) model, and analyze the possible solutions from an optimization perspective. Then, based on the analysis, we propose two effective algorithms to solve the problem, one named Sequential Fixing (SF), which provides near-optimal solutions in a short computing time, another based on a greedy approach that provides a

baseline for comparison. Table I summarizes the notation used throughout this section.

TABLE I: Summary of used notation.

Parameters	Definition
$\mathcal{N}$	Set of traffic types
$\mathcal{E}$	Set of edge clouds
$\mathcal{L}_i$	Set of links in the shortest path from the user to i
B_l	Bandwidth between two edge clouds connected with link l
C_i	Network capacity of edge network i
S_i	Computation capacity of edge cloud i
λ_n	User traffic rate for traffic type n
τ_n	Tolerable delay for serving the total traffic of type n
Variables	**Definition**
c_n	Slice of the network capacity for traffic type n
$b_{n,i}$	Indicator of whether traffic n is processed on node i
$\alpha_{n,i}$	Percentage of traffic n processed on node i
$\beta_{n,i}$	Percentage of computation capacity S_i sliced for traffic n

A. Mathematical Model

Our goal is to minimize the total latency of serving several types of user traffic in multiple edge networks with a certain network topology. This latency is the sum of the *wireless network latency* of the user ingress point and the *outsourcing latency* which, in turn, is composed of the *processing latency* in the edge clouds and the *link latency* between edge clouds.

Wireless Network Latency: We model the transmission of traffic in user ingress point as an $M/M/1$ processing queue. The *wireless network latency* for transmitting the user traffic of type n, denoted by t_n^W, can therefore be computed as:

$$t_n^W = \frac{1}{c_n - \lambda_n}, \ \forall n \in \mathcal{N}, \tag{1}$$

where c_n is the capacity of the network slice for traffic type n in the ingress edge network and λ_n is the traffic rate for n. The following constraints ensure that the capacity of all slices does not exceed the total capacity C_i of the ingress edge network, and c_n is higher than the corresponding λ_n:

$$\sum\nolimits_{n\in\mathcal{N}} c_n \leqslant C_i, \tag{2}$$

$$\lambda_n < c_n, \ \forall n \in \mathcal{N}. \tag{3}$$

Processing Latency: We assume that each type of traffic can be segmented and processed on different edge clouds, and each edge cloud can slice its computation capacity to serve different types of traffic. We denote with $\alpha_{n,i}$ the percentage of type n traffic processed on edge node i, and with $\beta_{n,i}$ the percentage of computation capacity S_i sliced for type n traffic. The processing of user traffic is described by an $M/M/1$ model. Let $t_{n,i}^P$ denote the *processing latency* of edge cloud i for user traffic n. Then, based on the computational capacity slice ($\beta_{n,i}S_i$) and the amount of traffic served on node i ($\alpha_{n,i}\lambda_n$), the processing latency $t_{n,i}^P, \forall (n,i) \in \mathcal{N} \times \mathcal{E}$ is expressed as follows:

$$t_{n,i}^P = \begin{cases} \frac{1}{\beta_{n,i}S_i - \alpha_{n,i}\lambda_n} & \text{if } \alpha_{n,i} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In the above equation, when traffic n is not processed on edge cloud i (i.e., $\alpha_{n,i} = 0$), the latency is 0; at the same time, no computation resource of i should be sliced to n (i.e., $\beta_{n,i} = 0$). The corresponding constraint is written as:

$$\begin{cases} \alpha_{n,i}\lambda_n < \beta_{n,i}S_i, & \text{if } \alpha_{n,i} > 0, \\ \alpha_{n,i} = \beta_{n,i} = 0, & \text{otherwise.} \end{cases} \tag{5}$$

$\alpha_{n,i}$ and $\beta_{n,i}$ also have to fulfill the consistency constraints:

$$\sum\nolimits_{i\in\mathcal{E}} \alpha_{n,i} = 1, \ \forall n \in \mathcal{N}, \tag{6}$$

$$\sum\nolimits_{n\in\mathcal{N}} \beta_{n,i} \leqslant 1, \ \forall i \in \mathcal{E}. \tag{7}$$

Link Latency: When each type of traffic is outsourced from user ingress point to edge cloud i, the transmission latency depends on $B_l, \forall l \in \mathcal{L}_i$, as well as on the total traffic passing through these links, denoted by F_l. To compute F_l, we first write the volume of traffic which flows to an edge network j as $\sum_{n\in\mathcal{N}} \alpha_{n,j}\lambda_n$. Then, $F_l, \forall l \in \mathcal{L} = \bigcup_{i\in\mathcal{E}} \mathcal{L}_i$, is written as:

$$F_l = \sum_{j\in\mathcal{E}, if\ l\in\mathcal{L}_j} \sum_{n\in\mathcal{N}} \alpha_{n,j}\lambda_n. \tag{8}$$

Let $t_{n,i}^L$ denote the *link latency*. When i is the user ingress point, denoted by $ingress\_point$, $\mathcal{L}_i = \varnothing$ and $t_{n,i}^L = 0$. $\forall (n,i) \in \mathcal{N} \times (\mathcal{E} - \{ingress\_point\})$, $t_{n,i}^L$ is defined as:

$$t_{n,i}^L = \begin{cases} \sum\limits_{l\in\mathcal{L}_i} \frac{1}{B_l - \sum\limits_{j\in\mathcal{E}, if\ l\in\mathcal{L}_j} \sum\limits_{n'\in\mathcal{N}} \alpha_{n',j}\lambda_{n'}}, & \text{if } \alpha_{n,i} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Like Eq. (4), the *link latency* is counted only if a certain traffic segment is processed on i. The constraint for F_l is written as:

$$F_l < B_l, \ \forall l \in \mathcal{L}. \tag{10}$$

Now we can define the *outsourcing latency* for traffic type n, which depends on the longest serving time among edge clouds:

$$t_n^{PL} = \max_{i\in\mathcal{E}} \{t_{n,i}^P + t_{n,i}^L\}, \ \forall n \in \mathcal{N}. \tag{11}$$

Optimization Problem - JSNC: Our problem is to minimize the total latency of network and computation slices under the constraint of tolerable delay for each type of traffic:

$$\min_{c_n, \alpha_{n,i}, \beta_{n,i}} \sum_{n\in\mathcal{N}} \{t_n^W + t_n^{PL}\}, \tag{$\mathcal{P}0$}$$

$$\text{s.t.} \quad t_n^W + t_n^{PL} \leqslant \tau_n, \ \forall n \in \mathcal{N}, \tag{12}$$

$$(2), (3), (5), (6), (7), (10).$$

Problem $\mathcal{P}0$ contains both nonlinear and indicator constraints, therefore, it is a mixed-integer nonlinear programming (MINLP) problem, which is hard to be solved directly [4].

Problem Reformulation: We now propose an equivalent reformulation of $\mathcal{P}0$, which can be solved very efficiently with the Branch and Bound method. Moreover, the reformulated problem can be further relaxed and, based on that, we propose our heuristic algorithms which can get near-optimal solutions in a short computing time. To this aim, we first introduce the binary variable $b_{n,i}$ to replace indicator $\alpha_{n,i} > 0$ in $\mathcal{P}0$, which

denotes whether traffic n is processed on i or not. Therefore we have:

$$\epsilon b_{n,i} \leqslant \alpha_{n,i} \leqslant b_{n,i},\ \forall (n,i) \in \mathcal{N} \times \mathcal{E}, \tag{13}$$

where $\epsilon > 0$ is a small value.
Then, Eq. (11) can be written as $t_n^{PL} \geqslant (t_{n,i}^P + t_{n,i}^L),\ \forall (n,i) \in \mathcal{N} \times \mathcal{E}$. Based on the definitions of $t_{n,i}^P$ and $t_{n,i}^L$, it can be rewritten as:

$$t_n^{PL} \geqslant \frac{1}{\beta_{n,i} S_i - \alpha_{n,i}\lambda_n} + \sum_{l \in \mathcal{L}_i} \frac{1}{B_l - \sum\limits_{j \in \mathcal{E}, if\ l \in \mathcal{L}_j} \sum\limits_{n' \in \mathcal{N}} \alpha_{n',j}\lambda_{n'}},$$
$$\forall (n,i) \in \mathcal{N} \times \mathcal{E} \mid b_{n,i} = 1. \tag{14}$$

Inequality (14) associates with $b_{n,i}$, which is an indicator variable. We observe that: if $b_{n,i} = 1$, we have $\frac{1}{\beta_{n,i} S_i - \alpha_{n,i}\lambda_n} > \frac{1}{S_i} \geqslant \frac{1}{\max_{j \in \mathcal{E}} S_j}$; otherwise $\beta_{n,i} S_i - \alpha_{n,i}\lambda_n = 0$ resulting in $t_n^{PL} \to \infty$. Therefore, we define a new variable $t_{n,i}^{P'}, \forall (n,i) \in \mathcal{N} \times \mathcal{E}$ to handle the above case:

$$t_{n,i}^{P'} = \frac{1}{\beta_{n,i} S_i - \alpha_{n,i}\lambda_n + \bar{b}_{n,i} S^{max}}, \tag{15}$$

where $\bar{b}_{n,i} = 1 - b_{n,i}$, and $S^{max} = \max_{j \in \mathcal{E}} S_j$. Using $b_{n,i}$, constraint (5) can be transformed into:

$$\alpha_{n,i}\lambda_n - \bar{b}_{n,i} < \beta_{n,i} S_i \leqslant b_{n,i} S_i,\ \forall (n,i) \in \mathcal{N} \times \mathcal{E}. \tag{16}$$

For $t_{n,i}^L$, we rewrite it as $t_{n,i}^L = b_{n,i} \sum_{l \in \mathcal{L}_i} v_l$, where

$$v_l = \frac{1}{B_l - F_l},\ \forall l \in \mathcal{L}. \tag{17}$$

Then, inequality (14) can be equivalently transformed into:

$$t_n^{PL} \geqslant t_{n,i}^{P'} + t_{n,i}^L,\ \forall (n,i) \in \mathcal{N} \times \mathcal{E}. \tag{18}$$

Now, we aim at linearizing $t_{n,i}^L = b_{n,i} \sum_{l \in \mathcal{L}_i} v_l$. We first compute the boundaries on v_l: $B_l^{-1} \leqslant v_l \leqslant V_l = \frac{1}{max\{\epsilon,\ B_l - \sum_{n' \in \mathcal{N}} \lambda_{n'}\}}$, where the $\max\{\cdot\}$ function permits to avoid the case of $B_l \leq \sum_{n' \in \mathcal{N}} \lambda_{n'}$. Based on above and constraint (12), $t_{n,i}^L$ is linearized:

$$b_{n,i} \sum\nolimits_{l \in \mathcal{L}_i} B_l^{-1} \leqslant t_{n,i}^L \leqslant b_{n,i} \tau_n, \tag{19}$$
$$\bar{b}_{n,i} \sum\nolimits_{l \in \mathcal{L}_i} B_l^{-1} \leqslant \sum\nolimits_{l \in \mathcal{L}_i} v_l - t_{n,i}^L \leqslant \bar{b}_{n,i} \sum\nolimits_{l \in \mathcal{L}_i} V_l. \tag{20}$$

Finally, the equivalent reformulation of $\mathcal{P}0$ can be written as:

$$\min_{\substack{c_n, \alpha_{n,i}, \beta_{n,i}, b_{n,i}, \\ t_n^W, t_n^{PL}, t_{n,i}^{P'}, t_{n,i}^L, v_l}} \sum_{n \in \mathcal{N}} (t_n^W + t_n^{PL}), \tag{$\mathcal{P}1$}$$
$$\text{s.t.} \quad (1), (2), (3), (6), (7), (10), (12), (13), (15), (16), (17), (18), (19), (20).$$

In problem $\mathcal{P}1$, $c_n, \alpha_{n,i}, \beta_{n,i}$ and $b_{n,i}$ are the main decision variables, while others are auxiliary. All the variables are bounded. Since constraints (1), (15) and (17) are quadratic while the others are linear, $\mathcal{P}1$ is a mixed-integer quadratically constrained programming (MIQCP) problem.

B. Heuristics: Sequential Fixing and Greedy approaches

Since solving problem $\mathcal{P}1$ can be still time-consuming, especially in medium to large-scale-scenarios, we further propose two effective heuristics, detailed hereafter: the first is a Sequential Fixing method, the second is a Greedy approach and is mainly proposed for reference, as a baseline approach. We will show in the numerical evaluation section that they both achieve very good (in most cases near-optimal) solutions in all the scenarios we considered, which are representative of typical edge clouds, in a short computing time.

1) Sequential Fixing (SF): This heuristic, detailed in Algorithm 1, is based on the following rationale: first, it solves a relaxation of the original problem (this step takes a very short time, typically less than 1 second even for the largest scenarios we evaluated). Then, based on the relaxed solution $\tilde{b}_{n,i}^*$, which can be seen as the probability of edge selection, edge nodes are ranked and selected (see lines 1-2). We also rank the traffic types, according to the rate and corresponding tolerable latency, and allocate computing nodes to the different types of traffic via a specific scheme (see lines 3-8). Finally, we eliminate all unfixed $b_{n,i}$ to further prune the problem.

Algorithm 1 *Sequential Fixing (SF)*

1: Relax $b_{n,i}$ to continuous $\tilde{b}_{n,i}$ in $\mathcal{P}1$, then solve the relaxed problem to obtain optimal relaxed values $\tilde{b}_{n,i}^*$;
2: Rank nodes in $\mathcal{E}$ by descending values of $\sum_{n \in \mathcal{N}} \tilde{b}_{n,i}$, and keep top $\mathcal{K} \subset \mathcal{E}$ (where $|\mathcal{K}| = 0.50 * |\mathcal{E}|$);
3: Rank traffic types in $\mathcal{N}$ in descending order (λ_n *first,* τ_n *second*);
4: *Allocate* nodes in $\mathcal{K}$ to ordered traffic types (*i.e., setting* $b_{n,i} = 1$) till either $\mathcal{K}$ or $\mathcal{N}$ is *completely scanned*;
5: **if** $|\mathcal{K}| < |\mathcal{N}|$ **then**
6: Rank $\mathcal{K}$ in descending order (S_i *first,* $|\mathcal{L}_i|$ *second*);
7: *Allocate* $\mathcal{K}$ to (*remaining* $\mathcal{N}$) repeatedly;
8: **else**: *Allocate* (*remaining* $\mathcal{K}$) to $\mathcal{N}$ repeatedly;
9: Set the remaining variables $b_{n,i} = 0$.

2) Greedy approach: The greedy approach, detailed in Algorithm 2, first ranks edge nodes according to the link and computation capacities (see line 1). Then, it ranks the traffic types and allocates edge nodes (with a fraction $\left\lceil \frac{\lambda_n |\mathcal{K}|}{\sum_{n' \in \mathcal{N}} \lambda_{n'}} \right\rceil$) to each type (see lines 2-5). Finally, it branches the problem in the same way as SF.

V. Numerical Results

In this section we evaluate and compare, in a set of typical mobile edge network scenarios, the proposed model (identified as $Optimal$ in the following) the two heuristics, SF and $Greedy$, and $Random$. This last one is used as reference and is based on randomly selecting nodes (by setting $b_{n,i}$; for each network instance we ran it for 1000 times and selected the best solution).

Experimental Setup: We implement our model and heuristics using SCIP (Solving Constraint Integer Programs)[1], an

[1] http://scip.zib.de/

Algorithm 2 *Greedy Approach*

1: Generate *priority order* of edge nodes $\mathcal{E}$ ($|\mathcal{L}_i|$ *first,* S_i *second*, by rotating the topology graph (see Figure 2) around the user ingress point and doing level traversal of the graph), keeping the top $\mathcal{K}$ nodes ($|\mathcal{K}| = 0.50 * |\mathcal{E}|$);
2: Rank traffic types $\mathcal{N}$ in descending order (λ_n *first,* τ_n *second*);
3: *Allocate* $\mathcal{K}$ to $\mathcal{N}$ ($\forall n \in \mathcal{N}$, $amount = \left\lceil \frac{\lambda_n |\mathcal{K}|}{\sum_{n' \in \mathcal{N}} \lambda_{n'}} \right\rceil$) in order, till either $\mathcal{K}$ or $\mathcal{N}$ is *completely scanned*;
4: **while** $\mathcal{N}$ has un-allocated elements **do**
5: *Re-allocate* $\mathcal{K}$ to (*rest of* $\mathcal{N}$) one by one repeatedly;
6: Set the remaining variables $b_{n,i} = 0$.

open-source framework that solves constraint integer programming problems. All numerical results presented in this section have been obtained on a server equipped with an Intel(R) Xeon(R) E5-2640 v4 CPU @ 2.40GHz and 126 Gbytes of RAM. The parameters of SCIP in our experiments are set as their default.

In our model we have 5 types of parameters: traffic λ_n, computation capacity S_i, network capacity C_i, link capacity B_l and tolerable latency τ_n for each type of traffic. Table II shows the initial values of all parameters. They are representative of a scenario with high traffic load and low tolerable latency relative to the limited communication and computation resources. The three types of traffic we considered can be interpreted straightforwardly as *social* (small traffic, low tolerable latency), *web* (medium traffic, medium tolerable latency) and *video* (high traffic, low tolerable latency). It is worth noticing that unit "cycles/s" is often used for computation capacity; for simplicity we transform it into "Gb/s" by using the factor "8bit/1900cycles", which assumes that processing 1 byte of data needs 1900 CPU cycles in a BBU pool [14].

We conduct our simulations by scaling one parameter value at the time, starting from the initial values in Table II, while fixing all other parameters. The total latency (the objective function value) and the model solving time are recorded and plotted in the figures illustrated hereafter.

Network Topologies: Figure 2 shows the four different topologies of the mobile edge network we selected for measuring the performance of our proposed approaches: 3 tree and 1 chain topologies. These are typically used for performance

TABLE II: Parameters setting - Initial values (for the case of high incoming traffic load with low tolerable latency)

Parameter	Initial value	
$\mathcal{N}$	$\{1, 2, 3\}$	
$\mathcal{E}$	$\{1, 2, \cdots, 15\}$	
$\mathcal{L}$	*Links in the graphs (see Figure 2)*	
B_l (Gb/s)	$72^{(\times 14)}$	$(l = 1, 2, \cdots, 14)$
C_i (Gb/s)	75	$(i = 15)$
S_i (Gb/s)	$30^{(\times 1)}, 25^{(\times 2)}, 20^{(\times 4)}, 15^{(\times 8)}$	$(i = 1, 2, \cdots, 15)$
λ_n (Gb/s)	15, 20, 35	$(n = 1, 2, 3)$
τ_n (ms)	1.0, 1.5, 1.0	$(n = 1, 2, 3)$

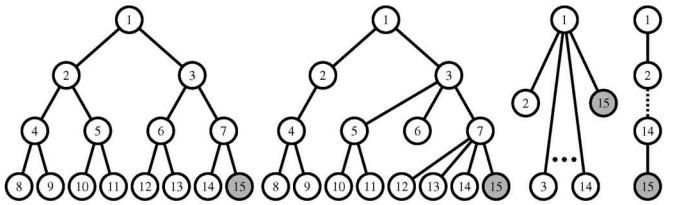

Fig. 2: Four selected network topologies (15 nodes).

assessment in MEC scenarios, which are similar to [15]. Each node in the figures represents the computing node and each edge between any two nodes represents a communication link. The user's ingress point for each case can be one of the leaf nodes in the bottom tier of graphs. For simplicity, we set node 15 (marked in light gray) as the user's ingress point for all 4 topologies in our simulations. All topologies have the same parameters settings. We underline that our model and heuristics are general, and can be applied in any topology with a predetermined routing among different nodes, which can be performed in a pre-processing phase.

Overview of the Evaluation: We investigate the effect of all parameters on the total latency (see Figures 3, 4, 5, 6, 7, and 8). In Figure 7, we investigate, in particular, the effect of the network topology and parameter scaling. In Figure 8, we further study a larger problem instance obtained by increasing the number of traffic types from 3 to 15 without changing the total amount of traffic (scaling both network capacity C_i and traffic rate λ_n to achieve feasible solutions). Finally, we compare the computing time of all 4 approaches in Figure 9.

A. Effect of the traffic λ_n

Figure 3 shows the total latency variation versus traffic rate under the case of $|\mathcal{N}| = 3, |\mathcal{E}| = 15$. Specifically, we scale λ_n, $\forall n \in \mathcal{N}$ from 0.1 to 1 based on the initial value in Table II, while fixing all other parameters. The total latency in all scenarios increase when the traffic raises and the increasing rates grow rapidly with the scale approaching to 1.0. Our proposed SF and $Greedy$ approaches have almost the same performance (SF being always better than $Greedy$), and they are very close to the optimum, especially when traffic scales above 0.5. The $Random$ approach exhibits a larger total latency with respect to the other allocation schemes, the difference being around 0.3 ms. All the curves have the shape of a typical flipped inverse-proportional function due to the expression of λ_n in the model objective function. Finally, we observe that all our approaches are characterized by smooth curves, including the $Optimal$ JSNC model, which indicates stability in the solving approach.

B. Effect of the tolerable latency τ_n

We now illustrate (Figure 4) the total latency with respect to the tolerable latency τ_n, with different scaling factors (from 0.70 to 0.86 w.r.t. the initial value). The vertical dashed lines represent the threshold for each approach, showing the minimum tolerable latency that can be requested in order to generate a feasible solution. The thresholds are $Optimal$ $(0.715) <$

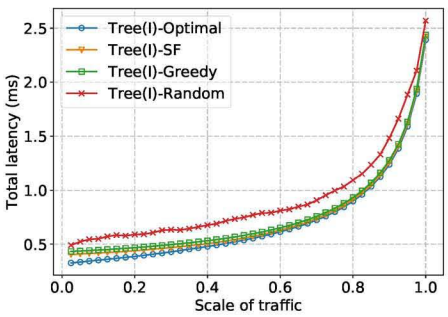

Fig. 3: Scaling λ_n ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$).

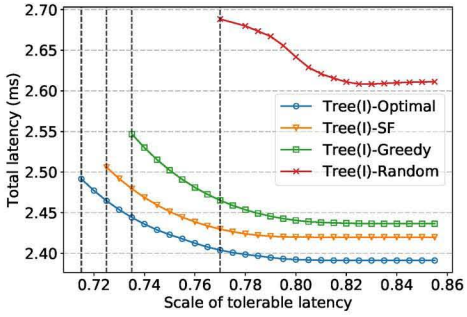

Fig. 4: Scaling τ_n ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$).

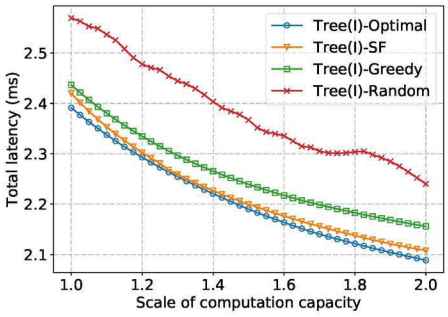

Fig. 5: Scaling S_i ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$).

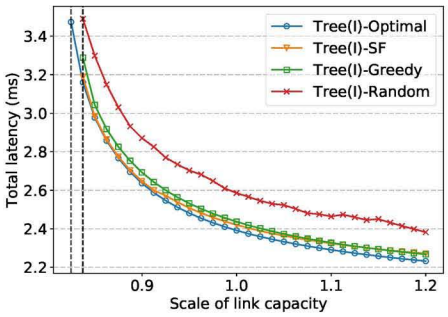

Fig. 6: Scaling B_l ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$).

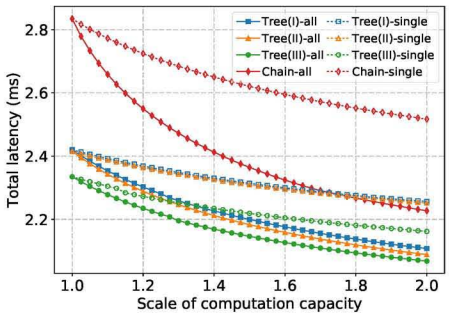

Fig. 7: Scaling S_i ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$) under 8 scenarios - SF heuristic.

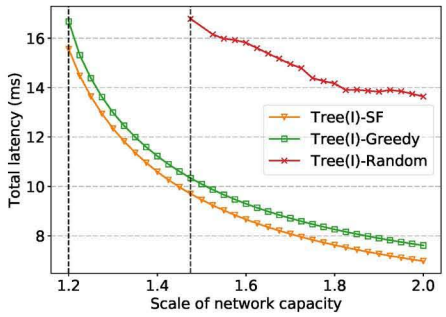

Fig. 8: Scaling C_i with a large number of traffic types ($|\mathcal{N}| = \mathbf{15}, |\mathcal{E}| = 15$).

$SF\,(0.725) < Greedy\,(0.735) < Random\,(0.77)$. When τ_n increases beyond the threshold, the total latency values under the $Optimal$, SF and $Greedy$ approaches decrease and converge to different values at almost the same point around 0.8 (for example, the tolerable latency of $Optimal$ decreases from 2.49 to 2.39). Parameter τ_n serves in our model as an upper bound (see constraint (12)), and limits the solution space. In fact, with a low τ_n value, the feasible solution set is smaller and the total latency increases, and vice versa.

We observe that the gap between the proposed Sequential Fixing (SF) approach and the $Optimal$ solution is indeed very small, around 1.43%.

C. Effect of the computation capacity S_i

Figure 5 shows the total latency as a function of the computation capacity $S_i, \forall i \in \mathcal{E}$, which is scaled from 1 to 2 w.r.t. its initial value. The total latency decreases when the computation capacity increases, which is also consistent with the real case. As S_i increases by a factor of two, the latency in all cases decreases of about 0.3 ms. Regarding the different approaches we considered, $Random$ performs the worst, SF is very close to $Optimal$, and practically overlapping in the (1.2, 1.5) range. The $Greedy$ algorithm also performs well, with a gap that reaches 1.76% w.r.t. to the $Optimal$ solution.

D. Effect of the link capacity B_l

Figure 6 illustrates the total latency variation as a function of the link capacity $B_l, \forall l \in \mathcal{L}$, which is scaled from 0.82 to 1.2 with respect to its initial value. Like in the case shown in Figure 4, here there also exist thresholds of the feasible solutions for scaling link capacity (0.825 for the $Optimal$ scheme, which has a maximum total latency of 3.47 ms, and 0.838 for all the other approaches). This is due to the fact that when the links' capacity is low, only part of traffic can be routed through them, hence making the local processing time become too long to satisfy the maximum tolerable latency for each traffic type. As B_l increases above the threshold, the total latency in all cases decreases and converges to different points (around 2.27 ms for $Optimal$, SF, $Greedy$, around 2.38 for $Random$). SF performs very close to $Optimal$, and overlaps with it in the (0.838, 0.92) range. The $Greedy$ approach also exhibits a good performance, with a gap of 1.44% w.r.t. the $Optimal$ solution.

E. Effect of the network topology and parameter scaling

Figure 7 illustrates the results achieved by SF when scaling the computation capacity S_i ($|\mathcal{N}| = 3, |\mathcal{E}| = 15$) in the 4 network topologies and two parameter scaling cases. Specifically, the solid lines show the results of scaling S_i, $\forall i \in \mathcal{E}$ (tag "all"), while the dashed lines stand for S_{15} (tag "single"), which is the user ingress node. Evidently, there are large performance gaps between scaling one node computation capacity and scaling all, and it can be observed that all tree topologies outperform the chain topology with a gap around 16.7%.

F. Effect of the network capacity C_i - Multiple traffic types

We now consider a scenario with a larger number of traffic types ($|\mathcal{N}| = 15$). Figure 8 illustrates the total latency with respect to the network capacity C_i (scaled from 1 to 2 w.r.t. to its initial value). Note that the results for $|\mathcal{N}| = 3$ exhibit a similar trend, and are not shown for the sake of space. Since obtaining the optimal solution in such large-scale scenario is

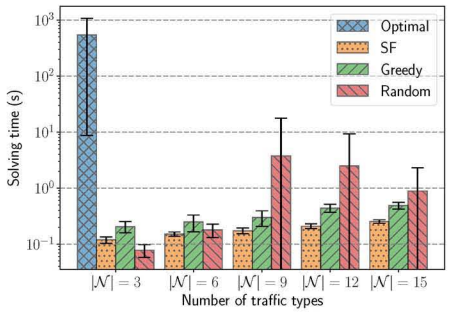

Fig. 9: Computing time (increasing $|\mathcal{N}|$ values, $|\mathcal{E}| = 15$).

impossible in a reasonable amount of time, we only show the performance of our proposed heuristics as well as the $Random$ approach. SF ensures the lowest latency, while $Greedy$ is 6.24% higher, in average. The total latency decreases from about 16 to 7 ms, while for $Random$, from approximately 17 to 14 ms. Our heuristics can thus handle large-scale problems and provide good solutions.

Computing time: We select one case (increasing the number of aggregated traffic types, $|\mathcal{N}|$, which increases the problem scale) to compare the computing time of the four approaches we considered in this paper (see Fig. 9). Obtaining the $optimal$ results takes a very long time (around 1000 seconds and beyond even for the smallest instances with $|\mathcal{N}| = 3$, while it was impossible for larger $|\mathcal{N}|$ values). Our proposed approaches are extremely fast in all considered network instances and topologies. With $|\mathcal{N}| = 3$, the solving time of SF is around 0.11s and $Greedy$ 0.18s. When $|\mathcal{N}|$ increases to 15, the solving time is still very low and less than 1s. This permits to perform in real-time network slice dimensioning and resource allocation, which is a key feature for providing the necessary QoS levels in next-generation mobile architectures.

VI. Conclusion

This paper proposed a novel mathematical model to perform a joint allocation of mobile network and edge computational resources. We investigated the problem by minimizing the total latency of multiple classes of user traffic with corresponding tolerable delay requirements, considering multiple edge networks in typical hierarchical network topologies. Furthermore, we transformed the original MINLP problem into an equivalent MIQCP one, and proposed two heuristics, i.e., Sequential Fixing (SF) and a greedy approach, based on the reformulation. Then we evaluated the performance of our model and heuristics, showing the effects of all considered parameters on the optimal and approximate solutions. Numerical results show that our SF approach can obtain near-optimal solutions in a very short computing time (around 0.11s) even for large-scale scenarios. When evaluating the maximum latency tolerated by user traffic, the performance of SF has a gap around 1.43% w.r.t. the optimal one, while in all other cases it is very close to the optimum. As for our greedy approach, it obtains slightly suboptimal solutions compared to SF (with a maximum gap of 1.76% w.r.t. the optimum), still in a short time (around 0.1s).

Acknowledgment

This research was supported by the H2020-MSCA-ITN-2016 SPOTLIGHT under grant agreement number 722788.

References

[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

[2] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung, "Network slicing based 5G and future mobile networks: mobility, resource management, and challenges," *IEEE Comm. Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

[5] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[6] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[7] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[8] P. Luong, F. Gagnon, C. Despins, and L.-N. Tran, "Joint virtual computing and radio resource allocation in limited fronthaul green C-RANs," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2602–2617, 2018.

[9] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Quality of Service (IWQoS), IEEE/ACM 25th International Symposium on*, IEEE, 2017, pp. 1–10.

[10] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.

[11] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems," in *IEEE Int. Conference on Communications (ICC)*, 2018, pp. 1–6.

[12] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G HetNets," in *IEEE Int. Conference on Communications (ICC)*, 2018, pp. 1–6.

[13] J. Li, A. Wu, S. Chu, T. Liu, and F. Shu, "Mobile edge computing for task offloading in small-cell networks via belief propagation," in *IEEE International Conference on Communications (ICC)*, IEEE, 2018, pp. 1–6.

[14] J. Tang, W. P. Tay, T. Q. Quek, and B. Liang, "System cost minimization in cloud RAN with limited fronthaul capacity," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3371–3384, 2017.

[15] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, IEEE, 2016, pp. 1–9.