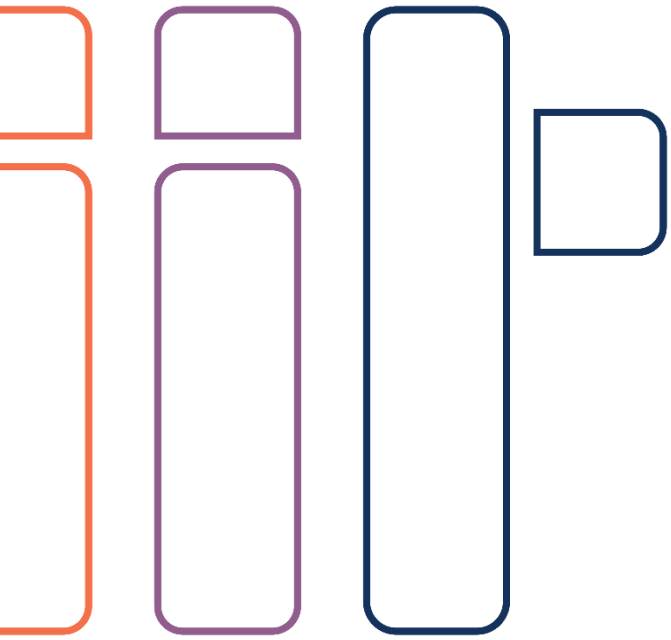

TECHNICAL REPORT

IIT TR-03/2021

Joint Device Association and Resource Allocation for Time-critical IoT Applications in MEC-empowered 5G Networks

S. Bolettieri, R. Bruno, E. Mingozzi

ISTITUTO DI INFORMATICA E TELEMATICA

Consiglio Nazionale delle Ricerche

Joint Device Association and Resource Allocation for Time-critical IoT Applications in MEC-empowered 5G Networks

Simone Bolettieri, Raffaele Bruno
Institute of Informatics and Telematics (IIT)
Italian National Research Council (CNR)
Pisa, ITALY
Email: {s.bolettieri,r.bruno}@iit.cnr.it

Enzo Mingozzi
Department of Information Engineering
The University of Pisa
Pisa, ITALY
Email: enzo.mingozzi@unipi.it

***Abstract*—Edge computing is emerging as an effective solution to fulfil the requirements of time-critical Internet of Things (IoT) applications by enabling the execution of data processing tasks at the network edge, in proximity to data producers. In particular, Multi-access edge computing (MEC) is attracting considerable interest in the mobile telco industry, as it supports edge computing capabilities within the mobile cellular network. However, to improve the network efficiency and QoS support of MEC systems, it is essential to jointly optimise edge resource management, IoT data collection and IoT device association in the context of heterogeneous services and edge resources. In this study, we address these challenges by first formulating the resource allocation, device association and data routing problem in a multi-cell MEC network as a mixed-integer non-linear programming problem that minimises the utilisation of communication and edge resources. We also propose a best-fit greedy heuristic method to determine an approximate solution to the optimisation problem for online resource management. Simulation results confirm the effectiveness of the proposed algorithm compared to three alternative benchmarks.**

***Index Terms*—mobile edge computing, IoT, service placement, device association, joint optimisation**

I. Introduction

In recent years, many time-critical IoT applications have emerged in the smart-city domain, which require the real-time processing of streams of sensed data from multiple sources. Typical examples are surveillance tasks where data from networks of sensors and cameras are analysed for scene reconstruction, human activity recognition or precise object tracking [1], [2].

Edge computing is proposed as a promising solution to satisfy the stringent low-latency requirements of these data-intensive IoT applications. It makes computation and storage capabilities available on edge devices in the access networks, thus helping in overcoming the fundamental latency limitations of centralised cloud systems [3]. In particular, Multi-access edge computing (MEC) is attracting considerable interest in the mobile telco industry, as (i) it supports the edge computing paradigm within the future 5G-based mobile networks [4], and (ii) it offers services that allow applications to exploit user proximity information (e.g., connection and location information of devices connected to a particular cell or access point) to improve QoS [5].

MEC servers (hosts) have more limited capacities than cloud counterparts. Thus, the design of optimised strategies to select where to execute services in a MEC network in order to both provide the best possible QoS for the user applications, and to accept as many requests as possible, has received a lot of attention [6]–[9]. However, prior work typically adopts a *user-oriented* perspective as the computing demands directly originate from the users' devices, which are also the producers of the data to be processed. On the contrary, the above-mentioned IoT applications require that a non-trivial amount of data is collected from dispersed IoT devices and then processed and analysed. Besides, multiple paths can be used to route the data traffic to the associated services, depending on: (i) the host on which each service is running, (ii) the set of IoT devices that are producing data for that service, and (iii) the particular base station with which such IoT devices are associated. There is only a handful of research studies that jointly consider the problem of device association and resource allocation in MEC networks. For instance, the authors in [10] address the problem of multi-task offloading and user association to minimise the total energy consumption of mobile users and MEC servers in a multi-cell network. A strategy for resource allocation and users' association is proposed in [11] to minimise the weighted sum of delays of both computing services and content delivery services. Content placement and user association in small cell networks is analysed in [12] to minimise the average content download delay. To the best of our knowledge, the problem of efficient resource utilisation with joint device association, data routing and service placement in a multi-cell MEC network for latency-sensitive IoT applications that collect data from multiple sources is still an open challenge. The key contributions of our work can be summarised as follows:

- We formulate an optimisation framework to jointly perform communication, computation and storage resource allocation, device association and data routing for time-critical IoT applications in multi-cell MEC networks aiming to minimise the network load and the utilisation of MEC resources. We

account for practical features of this system, such as overlapping coverage regions of base stations, inter-cell interference, and communication delays between MEC hosts.
- We develop a best-fit greedy heuristic method to determine a sub-optimal solution of the resource allocation problem for online resource management.
- We conduct an extensive simulation study to compare the efficacy of the proposed strategy with three benchmarks that perform device association and resource allocation independently. We show that our proposed strategy performs close-to-optimal.

II. System Model

In this section, we first introduce the model of our MEC-enabled system supporting delay-sensitive IoT monitoring and processing applications. Then, we present the communication and edge computing model.

A. System overview

We consider a heterogeneous cellular network consisting of one MBS and M SBSs underlying the same macrocell. Let $\mathcal{B} = \{b_o, b_1, \ldots, b_M\}$ denote the set of base stations, where b_0 is the MBS. Furthermore, we assume that there are N IoT devices (also denoted as UEs) in the macro cell, denoted as $\mathcal{U} = \{u_1, \ldots, u_N\}$. Each IoT device is equipped with a sensor that is used to collect measurements, e.g. about an environmental parameter, periodically. Let l_i denote the size (in bytes) of the data payload of a sensor measurement from device u_i. We also consider that an IoT device is restricted to be associated with one base station at any time, and $\mathcal{B}_i$ denotes the set of SBSs covering device u_i. Each base station is equipped with a MEC host (i.e. a server providing a virtualisation infrastructure) with heterogeneous computation and storage capability. Specifically, it is assumed that base station b_m has a storage capacity of Γ_m bytes and a computation capability Δ_m, measured in the number of CPU cycles per second [11]. Without loss of generality, we can also assume that the MBS hosts the MEC resource orchestrator that decides about resource allocation and association decisions between devices and base stations within the cell. Typically, each SBS is connected to the MBS via a backhaul link, which is either a physical link (wired or wireless) or a virtualised link created using the MEC-enabled virtualised network infrastructure [13].

In this study, we focus on a *class of delay-sensitive IoT applications that perform inference tasks over spatially dispersed sensor data streams periodically*. These applications typically require the periodic collection of sensory data from multiple sources, which need to be processed within a deadline, e.g. to be used as feedback in a control loop. Furthermore, we target periodic applications, namely the inference tasks are invoked with a fixed periodic schedule, such as in typical surveillance applications of physical phenomena or industrial monitoring applications [14]. More formally, a generic computation task s_j is described through a requirement vector $\langle \mathcal{U}_j, T_j^{max}, \delta_j, \gamma_j, \rangle$, where $\mathcal{U}_j \subseteq \mathcal{U}$ denotes the geographical scope of the application, i.e., the set of sensors producing the real-time data that is processed by the task; δ_j is the number of CPU cycles required to execute the inference tasks on one byte of input data; and γ_j is the storage space occupied by the persistent data associated with the task (e.g., code). We point out that the processing task of application j also needs additional storage space beyond γ_j to store the input data collected from sensors during each service execution period. Without loss of generality, we can make the simplifying assumption that the task execution deadline is equal to the periodicity of the computation process. In other words, the outcome of the computing tasks is due by the end of the service period. Finally, we assume that there is a set $\mathcal{S}$ of K independent computation tasks that need to be deployed on the MEC-enabled edge system to reduce the load in the mobile core network and to avoid the long network latencies to a remote centralised cloud. However, given the heterogeneous characteristics of the edge servers and the differences in available channel bandwidth between base stations, an appropriate strategy for resource allocation and device association is needed to ensure that the task execution constraints are fulfilled without wasting resources. We elaborate more on this in the following sections, wherein we describe the communication and computation models in detail.

For the sake of notation simplicity, in the following, we use the index i, j and m to denote device u_i, application/task s_j and baste station b_m, respectively, if no ambiguity occurs. Furthermore, we use m to refer to both the base station b_m and the MEC server hosted on that base station.

B. Communication model

As in [15], we assume that the bandwidth resources are orthogonal between the MBS and SBSs, and the overall uplink bandwidth of the MBS and SBSs is denoted as B_0^u and B_s^u, respectively. Furthermore, IoT devices that are associated with the same base station are also allocated orthogonal spectrum. Therefore, in this study, we only account for inter-cell interference between UEs associated with SBSs. Specifically, we define $x_i^m \in \{0, 1\}$ as the association decision variable, i.e., $x_i^m = 1$ if device $i \in \mathcal{U}$ is associated with base station $m \in \mathcal{B}$, and $x_i^m = 0$ otherwise. Furthermore, we define $\alpha_i^m \in [0, 1]$ as a decision variable that expresses the fraction of the uplink bandwidth of base station m that is assigned to UE i by the scheduler of that base station. Under an OFDMA-based access scheme, which allocates specific patterns of sub-carriers in the time-frequency space to different users, the instantaneous data rate for UE i is simply given by

$$R_i^m = B_s^u \cdot \eta_i^m \cdot \alpha_i^m \ , \forall m \in \{0, M\} \ , \tag{1}$$

where η_i^m is the spectrum efficiency of the uplink for UE i associated with base station m. If UE i is associated with SBS m ($m \neq 0$), the spectral efficiency of the uplink will be given by:

$$\eta_i^m = \log_2 \left(1 + \frac{P_i G_i^m}{\sigma_m^2 + \sum\limits_{\substack{i'=1 \\ i' \neq i}}^{N} \sum\limits_{\substack{m'=1 \\ i' \neq m}}^{M} \alpha_i^m \alpha_{i'}^{m'} P_{i'} G_{i'}^m} \right) \ , \tag{2}$$

where P_i is the transmission power of UE i, σ_m^2 is the power of the Gaussian noise at base station m, G_i^m is the channel

gain between UE i and base station m, and the summation is the inter-cell interference due to active IoT devices associated with other SBSs. For the sake of simplicity, we assume that radio access resources are assigned to devices in a random and independent way. Hence, $(\alpha_i^m \times \alpha_{i'}^{m'})$ represents the probability that two devices i and i' associated with two different SBSs share the same channel resources.

Similarly, if UE i is associated with the MBS, the spectral efficiency of the uplink is simply given by:

$$\eta_i^0 = \log_2\left(1 + \frac{P_i G_i^o}{\sigma_0^2}\right) , \tag{3}$$

as the uplink transmissions to the MBS are not affected by interference from the uplink transmissions to SBSs. It is important to point out that the average path loss between devices and SBSs is typically smaller than between devices and the MBS due to the proximity to SBSs. Finally, the complex interplay between scheduling decisions at SBSs (i.e.α_i^m) and inter-cell interference among SBSs introduces multiple non-linearities in the problem formulation. As discussed later, this dramatically affects the computational complexity of the problem.

C. Edge computing model

In our model, the computing task $j \in \mathcal{S}$ has to be executed within the T_j^{max} deadline. Let T_j^{total} denote the total execution time of task j. It must hold that $T_j^{total} \leq T_j^{max}$. It is straightforward to observe that the execution delay for a task j deployed on MEC server m consists of the processing latency, say T_j^{cpu}, and the total time that is needed to collect all the sensed input data generated by the devices in $\mathcal{U}_j$, say T_j^{data}. Thus, the completion time of the inference task can be expressed as

$$T_j^{total} = T_j^{cpu} + T_j^{data} , \tag{4}$$

We next discuss how to compute the T_j^{cpu} and T_j^{data} variables. For the sake of clarity Fig. 1 illustrates the different time components of T_j^{total}.

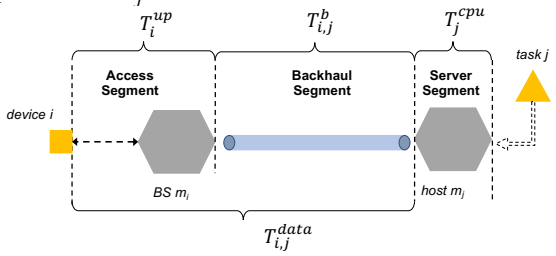

Fig. 1. Schematic diagram of the different components of the service execution latency.

1) MEC server processing latency: We let $y_j^m \in \{0, 1\}$ be the decision variable that indicates whether or not the MEC server hosted on base station m has to execute task j. Hence, y_j^m determines the service placement policy, which is essentially a rule for choosing which edge node to use for running the application. In this study, we assume an *elastic* computation model, i.e., a fraction $\beta_j^m \in [0, 1]$ of the computational capacity of base station m is allocated to task j. Then, the computation delay for the deployed task can be expressed as

$$T_j^{cpu} = \frac{\left(\sum_{i \in \mathcal{U}_j} l_i\right) \delta_j}{\beta_j^m \Delta_m} . \tag{5}$$

The above expression accounts for the total amount of input data (i.e., $\sum_{i \in \mathcal{U}_j} b_i$) that has to be processed by the task j during each execution period.

2) Data collection latency: To compute the overall data collection latency for task j, we introduce $T_{i,j}^{data}$, defined as the time that is needed by device i to transmit the input data of size l_i to the MEC server on which the task j is running. As shown in Fig. 1 this latency consists of two components. The first one is T_i^{up}, defined as the time to transmit the input data from the source device i to the base station with which the device is associated. This transmission latency depends on the uplink transmission rate of device i, which is given by

$$T_i^{up} = \sum_{m \in \mathcal{B}} x_i^m \frac{l_i}{R_i^m} . \tag{6}$$

The second component of the transmission latency is denoted as T_{m_i,m_j}^b, and it expresses the time needed to transmit the input data from the base station with which device i is associated (say m_i) and the MEC server that runs application j (say m_j). For the sake of generality, we assume that two MEC hosts are connected through a virtual data path p_{m_i,m_j}, which is established by the MEC virtualisation infrastructure to ensure guaranteed transmission delays. Specifically, path p_{m_i,m_j} can be modelled as a set $\mathcal{H}_{m_i,m_j}$ of virtual links, and the k-th link in this set introduces a fixed transmission delay equal to ω_{l_k}. In summary, the delay of collecting the input data for task j from device i is given as follows

$$\begin{aligned} T_{i,j}^{data} &= T_i^{up} + T_{m_i,m_j}^b \\ &= T_i^{up} + \sum_{m_i \in \mathcal{B}} x_i^{m_i} \sum_{m_j \in \mathcal{B}} y_i^{m_j} \sum_{l \in \mathcal{H}_{m_i,m_j}} \omega_l . \end{aligned} \tag{7}$$

We remind that we consider inference tasks that require data from multiple sensors that can be spatially dispersed within the cell. Since we assume that the data processing task of an application can only start when the MEC server executing that task has received all the input data, it holds that the data collection delay is determined by the maximal value of the transmission latencies of different data sources. Thus, it holds that

$$T_j^{data} = \max_{i \in \mathcal{U}_j} \left(T_{i,j}^{data}\right) . \tag{8}$$

It is important to observe that the same device i can generate data for multiple applications that are deployed on different MEC hosts. In this study we assume that each sensor produces a single data stream with a fixed data rate that is sufficient to meet the latency requirements of the most time-critical service. Then, the MEC host of the base station with which device i is associated, acts as a *data proxy* that replicates the same data to the other MEC servers requiring that input data. This design approach not only reduces the traffic to be transmitted on the uplink but also allows us to compute the T_j^{data} value independently for each application.

III. Joint Device Association and Edge Resource Allocation: Problem Formulation

In this section, we formulate the device association and edge resource allocation as a multi-objective optimisation problem.

Specifically, the optimisation function aims at minimising the resource consumption while finding a feasible resource allocation and association assignment that fulfils the latency requirements of requesting services.

First of all, let us define $\mathbf{x} = \{x_i^m\}$ as the vector of device association decisions; $\mathbf{y} = \{y_j^m\}$ as the vector of application placement decisions; $\boldsymbol{\alpha} = \{\alpha_i^m\}$ as the vector of data rate allocation; and $\boldsymbol{\beta} = \{\beta_j^m\}$ as the vector of computation resource allocation. Then, the total data traffic that is distributed among the MEC servers is given by

$$F(\mathbf{x},\mathbf{y}) = \sum_{i\in\mathcal{U}}\sum_{m_d\in\mathcal{B}} x_i^{m_d} \sum_{j\in\mathcal{S}}\sum_{m_s\in\mathcal{B}} y_j^{m_s}\left(b_i \cdot \left|\mathcal{H}_{m_{di},m_{sj}}\right|\right) \tag{9}$$

The rationale behind (9) is that the data generated by a device i and transmitted over the path p to input the processing task j has to be replicated on each virtual link of the backhaul network path $p_{m_{di},m_{sj}}$. In addition to the backhaul traffic $F(\mathbf{x},\mathbf{y})$, we also measure the resource consumption of the whole system in terms of $S(\mathbf{x},\boldsymbol{\alpha})$, defined as the total normalised spectrum utilisation, and $C(\mathbf{y},\boldsymbol{\beta})$ defined as the total normalised utilisation of the edge computing capacity. By definition, it holds that

$$S(\mathbf{x},\boldsymbol{\alpha}) = \frac{\sum_{i\in\mathcal{U}}\left(\sum_{m=1}^{M}\alpha_i^m\cdot B_s^u + \alpha_i^0\cdot B_0^u\right)}{M\cdot B_s^u + B_0^u}, \tag{10}$$

and

$$C(\mathbf{y},\boldsymbol{\beta}) = \frac{\sum_{j\in\mathcal{S}}\sum_{m\in\mathcal{B}}\beta_i^m\cdot\Delta_m}{\sum_{m\in\mathcal{B}}\Delta_m}, \tag{11}$$

Now, we formulate the optimisation problem as follows:

$$\begin{aligned}
\min\quad & \mu F(\mathbf{x},\mathbf{y}) + S(\mathbf{x},\boldsymbol{\alpha}) + C(\mathbf{y},\boldsymbol{\beta}) && && \text{(12a)}\\
s.t.\quad & T_j^{total} \le T_j^{max}, && \forall j\in\mathcal{S} && \text{(12b)}\\
& \sum_{m\in\mathcal{B}} x_i^m = 1, && \forall i\in\mathcal{U} && \text{(12c)}\\
& \sum_{m\in\mathcal{B}} y_j^m = 1, && \forall j\in\mathcal{S} && \text{(12d)}\\
& \sum_{j\in\mathcal{S}} y_j^m\left(\gamma_j + l_i\sum_{i\in\mathcal{U}_j} l_i\right) \le \Gamma_m, && \forall m\in\mathcal{B} && \text{(12e)}\\
& \sum_{i\in\mathcal{U}} \alpha_i^m \le 1, && \forall m\in\mathcal{B} && \text{(12f)}\\
& \sum_{i\in\mathcal{U}} \beta_i^m \le 1, && \forall m\in\mathcal{B} && \text{(12g)}
\end{aligned}$$

The optimisation function is a combination of the three objective functions (9), (10) and(11). The weighting coefficient μ is needed to ensure a proper scaling of the $F(\mathbf{x},\mathbf{y})$ function, and to avoid that solutions with low computation and storage costs but high backhaul traffic are not considered. Constraint (12b) guarantees that the service execution latency is bounded by the execution deadline. Constraints (12c) and (12d) ensure that in a feasible solution a device is associated with only one base station, and an application is deployed on only one MEC server, respectively. Constraint (12e) guarantees that the total amount of service data placed in a MEC server does not exceed its storage capacity. Constraint (12f) ensures that the total bandwidth resources assigned to devices associated with a base station do not exceed the available bandwidth of that base station. Similarly, constraint (12g) ensures that the total computation resources assigned to the tasks running on a MEC server do not exceed the overall computation capacity of that server.

Model linearisation: Problem defined in (12) is a non-convex mixed-integer non-linear programming (MINLP) problem, which is NP-hard and difficult to solve even for small problem sizes. However, it can be reformulated into a MILP problem by applying both exact and approximated linearisation techniques [16]. Due to space constraints, we do not discuss in detail the linearisation procedure for the problem in (12), but we sketch the line of reasoning we follow. First of all, we approximate the logarithm function in equation (2) using a *multivariate* linear regression model. The training data points are selected on a regular grid of α_i^m values. In order to reduce computational effort, we reduce the number of independent variables in the model by assuming that two UEs interfere only if they are sufficiently close. Now, we linearise the service completion time in (4). First, we observe that equation (5) is a nonlinear term because the decision variable β_j^m is at the denominator. However, to linearise this fraction, we can substitute β_j^m with L auxiliary binary variables, each representing a discretisation step for β_j^m. Then, we leverage these auxiliary variables to move the original decision variable to the numerator of the fraction. A similar non-linearity condition holds for equation (7). Finally, equation (8) in nonlinear as it includes a max function, but a linearisation transformation of the max function exists that requires the use of $|\mathcal{S}|$ additional continuous variables. It is important to point out that our relaxed version of Problem (12) decreases the computational difficulty of solving the problem. However, the optimal solution in the relaxed problem may be sub-optimal. Furthermore, the linearisation transformations come at the cost of introducing several new auxiliary variables, thus increasing the size of the state space.

IV. A Best-Fit Greedy Heuristic

As pointed out previously, Problem (12) is NP-Hard, and it can be solved in reasonable time only for small problem instances. To support online decision making, we also propose a best-fit greedy heuristic (called *BFG*) with backtracking that determines a sub-optimal solution of the original problem. The pseudo-code of BFG is reported in Algorithm 1.

BFG defines an order relationship among the tasks to deploy. Then, it greedily moves among the configurations that minimise the objective function (12a). Specifically, the heuristic creates an ordered list $\widehat{\mathcal{S}}$ of tasks from set $\mathcal{S}$ using the Earliest Deadline First (EDF) scheduling method [17] (line 2). In other words, BFG gives the highest priority to the task whose deadline is the closest. Then, the heuristic enters a loop by testing the deployment of each service $j' \in \widehat{\mathcal{S}}$ following the priority order (line 4). First, it creates a set $\widehat{\mathcal{B}}_{j'}$ (line 5) with all the base stations that have sufficient available storage resources to execute task j' (i.e., satisfying constraint (12e)). Now, the heuristic iterates among the base stations in $\widehat{\mathcal{B}}_{j'}$ to discover a feasible device assignment and resource allocation with the lowest value of the objective function. Specifically, at each

Algorithm 1 - BFG Algorithm.

Input: $\mathcal{U}, \mathcal{B}, \mathcal{S}$
Output: $vars = \{\mathbf{x}^*, \mathbf{y}^*, \alpha^*, \beta^*\}$ ▷ Optimal set of decision variables

1: $\Omega \leftarrow \{\emptyset\}$ ▷ Initialise the set of allocated services
2: $\widehat{\mathcal{S}} \leftarrow \texttt{SORT}(\mathcal{S})$ ▷ increasing order of T_j^{max} $(j \in \mathcal{S})$
3: **while** $(\widehat{\mathcal{S}} \neq \{\emptyset\})$ **do**
4: $j' \leftarrow \texttt{list}(\widehat{\mathcal{S}}).\texttt{pop}()$
5: $\widehat{\mathcal{B}}_{j'} :=$ set of $m \in \mathcal{B}$ satisfying (12e)
6: $vars_c \leftarrow vars$ ▷ shadow copy of current decision variables
7: **for all** $(m' \in \widehat{\mathcal{B}}_{j'})$ **do** ▷ Greedy search
8: $vars \leftarrow vars_c$
9: $\widehat{\mathcal{U}}_{j'} \leftarrow \texttt{SORT}(\mathcal{U}_{j'})$ ▷ decreasing order of $R_{i,max}$ $(i \in \mathcal{U}_{j'})$
10: $coverage_{j'} \leftarrow |\widehat{\mathcal{U}}_{j'}|$ ▷ # points to cover
11: **while** $(\widehat{\mathcal{U}}_{j'} \neq \{\emptyset\})$ **do**
12: $i' \leftarrow \texttt{list}(\widehat{\mathcal{U}}_{j'}).\texttt{pop}()$
13: $\widetilde{\mathcal{B}}_{i'} :=$ set of $m \in \mathcal{B}_{i'}$, eligible for i' association
14: $\widetilde{\mathcal{B}}_{i'} \leftarrow \texttt{SORT}(\widetilde{\mathcal{B}}_{i'})$ ▷ decreasing order of backhaul traffic $F(\mathbf{x}, \mathbf{y})$
15: **while** $(\widehat{\mathcal{B}}_{i'} \neq \{\emptyset\})$ **do**
16: $m'' \leftarrow \texttt{list}(\widehat{\mathcal{B}}_{i'}).\texttt{pop}()$
17: **if** $(\textsc{AllocRes}(i', j', m', m'') = \texttt{True})$ **then**
18: $coverage_{j'} \leftarrow coverage_{j'} - 1$
19: **break** ▷ Test next i'
20: **end if**
21: **end while**
22: **end while**
23: **if** $(coverage_{j'} > 0)$ **then**
24: **continue** ▷ Test next $m' \in \widehat{\mathcal{B}}_{j'}$
25: **else**
26: $vars_{j'}[].\texttt{push}(vars)$
27: **end if**
28: **end for**
29: $n :=$ element in $vars_{j'}[]$ that minimises (12a)
30: **if** $(\textsc{BackTrace}(vars_{j'}[n]) = \texttt{True})$ **then**
31: $vars \leftarrow vars_{j'}[n]$ ▷ Commit Allocation
32: **end if**
33: **end while**

iteration, a new base station $m' \in \widehat{\mathcal{B}}_{j'}$ is selected and a greedy search is started (line 7), using the same set of consolidated decision variables (line 8). The goal of this greedy search is to iterate among the IoT devices in $\mathcal{U}_{j'}$ to decide a device assignment and resource allocation that do not violate the execution deadline of task j'. More precisely, for each $i \in \mathcal{U}_{j'}$ BFG computes the maximum data rate, say $R_{i,max}$, that device i could obtain from the base stations it can associate to[1]. Then, BFG creates an ordered list $\widehat{\mathcal{U}}_{j'}$ of devices from set $\mathcal{U}_{j'}$ given the highest priority to the device with the lowest $R_{i,max}$ (line 9). Now, BFG iterates among the devices in $\widehat{\mathcal{U}}_{j'}$ (line 12). First, it creates the set $\widehat{\mathcal{B}}_{i'}$ of available base stations with which device i' can be associated (line 13). Clearly, if device i' is already associated with base station m from a previous BFG iteration, it holds that $\widehat{\mathcal{B}}_{i'} = \{m\}$, otherwise this set consists of all base stations of set $\mathcal{B}_{i'}$ with available bandwidth resources. Then, set $\widehat{\mathcal{B}}_{i'}$ is sorted into set $\widehat{B}_{i'}$. The ordering criterion is the amount of back-haul traffic that would be transmitted over the shortest path between $m'' \in \mathcal{B}_{i'}$ and m' (line 14). Now, BFG iterates among the set $\widehat{B}_{i'}$ (line 16) and it calls the function $\texttt{AllocRes}(i', j', m', m'')$. Due to space limitations we do not provide the pseudo-code description of this function but we summarise its operations. First, this function computes the amount of bandwidth and computing resources that are still available in $m'' \in \widehat{\mathcal{B}}_{i'}$ and $m' \in \widehat{\mathcal{B}}_{j'}$, respectively, and which could be assigned to device i' and task j', respectively. These values are given by:

$$\beta_{j',max}^{m'} = 1 - \sum_{l \in \mathcal{S} \setminus \{j'\}} \beta_l^{m'} \ , \tag{13a}$$

$$\alpha_{i',max}^{m''} = 1 - \sum_{l \in \mathcal{U} \setminus \{i'\}} \alpha_l^{m''} \ . \tag{13b}$$

It is intuitive to observe that if all the available bandwidth and computing resources of base station m'' and m' are used, the total service execution delay of j' would be minimised. However, Problem (12) does not aim at minimising the service latency but only to ensure that the service execution deadline is not violated. Since a fixed latency $T_{m'',m'}^b$ is accumulated on the backhaul data path between m'' and m', the latency requirement is satisfied when the sum of the uplink transmission delay and the server processing latency fulfils the following condition:

$$T_{i'}^{up} + T_{j'}^{cpu} \leq T_{j'}^{max} - T_{m'',m'}^b = T_{i',j'}^{target} \ . \tag{14}$$

The key question to address is how to split the latency budget $T_{i',j'}^{target}$ between the access segment and the edge segment (see Fig. 1). Clearly, a fixed splitting would be inefficient. Thus, BFG dynamically assigns a weight to the access and edge segments considering the current utilisation of bandwidth and computing resources. Specifically, let us introduce the weight factor $\widehat{\Theta}_{i',j'}^{m',m''}$ defined as

$$\widehat{\Theta}_{i',j'}^{m',m''} = \frac{T_{i'}^{up,min}}{T_{i'}^{up.min} + T_{j'}^{cpu.min}} \ , \tag{15}$$

$T_{i'}^{up.min}$ and $T_{j'}^{cpu.min}$ are the minimum uplink transmission delay and service execution delay, respectively, that would be obtained using the allocations of (13a) and (13b). Then, BFG computes a tentative allocation as follows. If device i' has been already associated with a base station in a previous BFG iteration, we define $T_{i'}^{up,cur}$ the current uplink transmission, otherwise we set $T_{i'}^{up,cur}$ to a very large value. Then, the target uplink transmission delay is given by

$$T_{i'}^{up.target} = \min\left\{ (1 - \widehat{\Theta}_{i',j'}^{m',m''})\, T_{i',j'}^{target}; T_{i'}^{up,cur} \right\} \ . \tag{16}$$

Similarly, let $T_{j'}^{cpu.cur}$ be the current service execution delay for service j' that has been computed in previous BFG iterations. Then, the target service execution delay is given by

$$T_{j'}^{cpu.target} = \min\left\{ T_{i',j'}^{target} - T_{i'}^{up.target}; T_{j'}^{cpu.cur} \right\} \tag{17}$$

Finally, the allocation function computes the pair of resource allocation variables $(\widehat{\alpha}_{i'}^{m''}, \widehat{\beta}_{j'}^{m'})$ that correspond to the delays

[1]The achievable data rate for device i associated with base station m is estimated by assuming that all the remaining bandwidth of m is assigned to device i.

in equation (17) and (16). If it holds that ($\widehat{\alpha}_{i'}^{m''} \leq \alpha_{i',max}^{m''}$ and $\widehat{\beta}_{j'}^{m'}) \leq \beta_{j',max}^{m'}$, the resource allocation is feasible and a new device in $\widehat{\mathcal{B}}_{i'}$ is tested if available (line 12). When all $i' \in \mathcal{U}_{j'}$ have been tested, BFG checks if they were all successfully covered. If yes, the new decision variables are saved (line 26).

At the end of the greedy search (line 28), BFG selects the tentative set of decision variables that minimises the objective function (12a) (line 29). Then, the `Backtrace()` function is invoked to check if deploying service j' does not violate the constraints of already deployed applications. Indeed, the activation of new devices changes the mutual interference between base stations, which may result in a decrease in effective transmission capacity. Thus, BFG checks if the eventual decrease in the transmission rates of already activated devices may be compensated by increasing the computation capacity assigned to the already deployed services so as to reduce the processing delays. If this compensation is feasible, the new decision variables are finally accepted (line 31).

V. Performance Evaluation

We consider a small-scale network topology of 500 m $\times$ 500 m consisting of one MBS, three SBSs, and 180 IoT devices. The MBS is located at the centre of the cell, while SBS and IoT devices are randomly distributed. However, we force a minimum distance between base stations and devices equal to 200 and 20 metres, respectively, to ensure a homogeneous deployment. To model the channel gain, we use the typical urban channel model defined in 3GPP standardisation [18]. Specifically, the path loss between the MBS and the UEs is expressed as $128.1 + 37.6log10(R)$. Similarly, the path loss between an SBS and the UEs is expressed as $140.7 + 36.7log10(R)$. Since device association and service deployment occur on relatively long timescales, we can neglect the impact of shadowing and fading on the average channel gain. Finally, the noise power is $\sigma^2 = 10^-11$ mW and the interference threshold is $I = -90$ dBm. The bandwidth of MBS and SBS are both equal to 10 MHz. We also assume that all SBSs are directly connected to the MBS with a wired backhaul link that introduces a fixed delay equal to 5 ms.

As a reference scenario, we focus on latency-sensitive applications that require the collection of high-resolution video frames from multiple sources to perform a video analytic task (e.g. behaviour analysis, moving object classification, object detection, object tracking). We assume that the size of a video frame is 512 KB, the computation is set within $[50, 100]$ cycles per bit of the uploaded data, and the storage footprint of the application is within $[2, 10]$ GB. If not otherwise stated, each application requires data from three cameras. Finally, the service execution deadline is set within $[5, 10]$ seconds.

A. Benchmarks

To verify the performance of our proposed solutions, we introduce the following three benchmark policies that decouple the device assignment problem from the resource allocation problem.

- *Without SBS* (WSBS): All devices are associated with the MBS, and all services are deployed on the MBS, following the same priority order of BFG. After the device assignment, the vector of data rate allocations is computed so as to assign the same share of B_0^u to active UEs. Finally, a closed-form solution for the allocation of computing resources is obtained using constraint (12b).
- *Nearest Assignment* (NEAS): Devices are associated with the base station with the lowest SNR (i.e., neglecting interference in (2)), while all services are deployed on the MBS. The vector of data rate allocations and compute resource allocations are computed using the same approach as WSBS. The only difference is that a fair bandwidth allocation is enforced at each base station independently.
- NEAS$^+$: Enhanced version of NEAS in which each service j is deployed on the base station that has connected the largest fraction of devices in set $\mathcal{U}_j$.

The rationale behind the two NEAS-based policies is to assess the advantage of using SBSs to collect the data that is produced by IoT devices. NEAS$^+$ is introduced to investigate the benefit of using distributed service deployment to mitigate the transmission delays on the MEC network.

All the following results are obtained by replicating 25 times each experiment with random service-request patterns. Average values and 95% confidence values are shown.

	WEAS	NEAS	NEAS+	BFG	OPT
Γ_0	600	600	300	300	300
$\Gamma_1, \Gamma_2, \Gamma_3$	0	0	100	100	100

TABLE I
DISTRIBUTION OF THE STORAGE CAPACITY AMONG BASE STATIONS. IN ALL THE CONSIDERED SCENARIOS THE TOTAL STORAGE CAPACITY OF THE MEC SYSTEM IS 600 GB

	WEAS	NEAS	NEAS+	BFG	OPT
Δ_0	10	10	5	5	5
$\Delta_1, \Delta_2, \Delta_3$	0	0	1.66	1.66	1.66

TABLE II
DISTRIBUTION OF THE COMPUTATION CAPACITY AMONG BASE STATIONS. IN ALL THE CONSIDERED SCENARIOS THE TOTAL COMPUTATION CAPACITY OF THE MEC SYSTEM IS 10 GHZ

B. Numerical results

In Fig. 3 we first explore the efficacy of each scheme by measuring the percentage of applications that can be successfully admitted. Table I and II we list the computation and storage capacity of MBS and SBS we use in the following experiments. It is important to note that to ensure a fair comparison between WSBS, NEAS and the other schemes, the computation and storage capacities of the MBS in WSBS and NEAS are set equal to the total amount of computation and storage resources that are available in the MEC network in the other scenarios. First, we can observe the optimal scheme is able to find a feasible service deployment in all the considered settings. On the contrary, WSBS can satisfy the constraint on service execution delays only for ten requesting applications, while its effectiveness rapidly degrades as the number of requesting applications increases. The other two benchmarks, NEAS and NEAS$^+$, are able to admit more applications than WSBS. However, our proposed heuristic outperforms all the

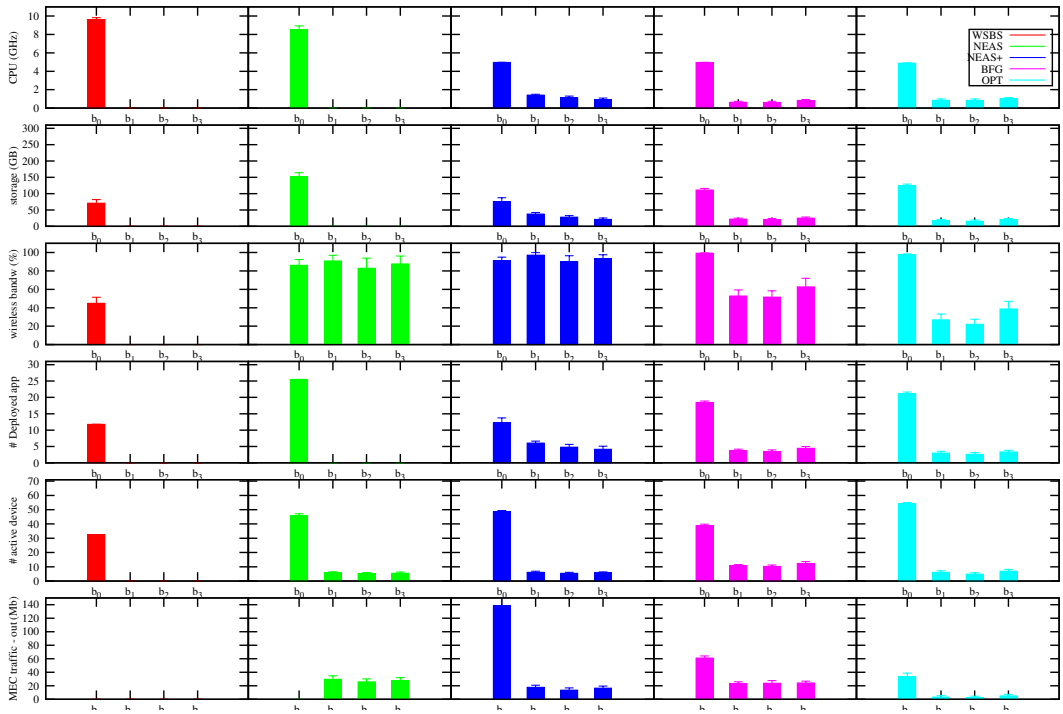

Fig. 2. Resource usage on each base station (B_0 is the MBS) for 30 requesting applications.

benchmarks, and it closely approximates the efficacy of the optimal scheme.

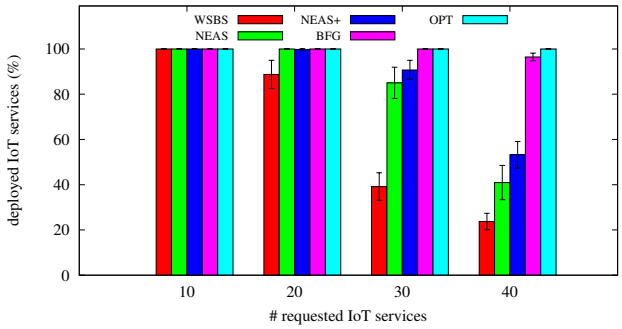

Fig. 3. Percentage of satisfied applications versus the number of requesting services.

To explain the root causes of the different efficacy of each scheme, Fig. 2 show how the network and edge resources are allocated to each base station in the case of 30 requesting applications. By design, 100% of deployed applications are running on the MBS, and 100% of activated IoT devices are associated with the MBS when using the WSBS scheme. Furthermore, each associated IoT device is assigned an equal share of the wireless bandwidth of the MBS. As shown in Fig. 2 (red bars), this approach allows the MEC system to fulfil the requirement of only 12 applications, even though there are unused bandwidth and storage resources. The reason is that the already admitted applications should share the computation resources of the MBS (which are 100% used) with each newly deployed service (we remind that we assume an elastic computing model), thus increasing their task processing delays. This would rapidly lead to the violation of the latency constraints of running services. On the contrary, IoT devices are also associated with SBSs when using the NEAS scheme (green bars). This generally results in lower uplink transmission delays than the ones observed with WSBS, since a higher percentage of available wireless bandwidth can be used. Thus, the same computation capacity at the MBS now allows the MEC system to accept up to 25 requesting applications. It is also interesting to note that about 20 Mbps of data traffic are transmitted from the SBSs to the MBS, where all the services are running. Clearly, there is a trade-off between the additional transmission delay that is accumulated on the backhaul links connecting SBSs to the MBS, and the more efficient utilisation of the available wireless bandwidth. The only difference between NEAS and NEAS+ (blue bars) is that in the latter scheme services can also be deployed on SBSs. We can observe that considerable data traffic is transmitted from the MBS to the SBS in NEAS+. This traffic is mainly due to data forwarded between SBSs. In these settings, the decentralisation of the storage and compute resources provides a small improvement on the number of admitted services (around 10%). It is also important to point out that in all the benchmarks, device association and bandwidth assignment are performed independently of service deployment, and according to a greedy approach. Each running service is allocated a set of resources that guarantees to fulfil constraint (12b). As a consequence, when the deployment of a new service fails, there is no advantage to reallocate the bandwidth resources that were initially assigned to the devices assigned to that services to the other running services. This also explains why there are unused bandwidth resources, even

if there are rejected services. Finally, the results in Fig. 2 show that our proposed BFG scheme (magenta bars) is able to reasonable mimic the allocation decisions of the optimal solution (cyan bars). In particular, the services are deployed in such a way to reduce the total traffic that is transmitted over the backhaul links, which is beneficial for reducing the overall data collection delays. Furthermore, the approach used by BFG to split the delay budget of a task between the access and edge segments effectively allows to balance computation and access resource utilisation.

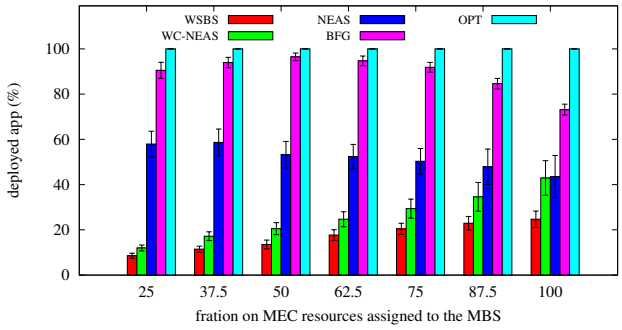

Fig. 4. Percentage of satisfied applications for different distributions of the resources between the MBS and SBSs. The results refer to a scenario with 40 requesting applications.

To investigate more in-depth the impact of resource decentralisation on the efficiency of the different algorithms, we conduct a second set of experiments in which a varying percentage of the total edge resources is assigned to the MBS. More precisely, 4 shows the fraction of satisfied service requests when the computation and storage capacity of the MBS is a percentage τ of the total MEC resources. When $\tau = 50\%$, we have the same settings of the previous experiments, $\tau = 25\%$ means that each base station has the same computation and storage capacity, and $\tau = 100\%$ means that SBSs do not have computation and storage capabilities. As expected, we observe that the WSBS efficacy improves with the increase of edge resources at the MBS. A four-fold increase of the computation and storage capacity of the MBS improves the number of satisfied services by three times. A similar trend is also observed with NEAS. On the contrary, when $\tau = 25\%$ NEAS+ admits 57.9% of the requesting applications against 8.4% of WSBS and 11.9% of NEAS. However, the NEAS+ performance converges to the one of NEAS as more edge resources are deployed in the MBS. Finally, BFG is far better than the other methods, and its effectiveness is less affected by the different τ values. However, the highest efficiency is obtained when $\tau = 50\%$, while a performance degradation is observed when the MEC resources are concentrated only on the MBS.

VI. Conclusions

In this paper, we studied resource allocation, data routing and device association in MEC-enabled multi-cell networks for delay-sensitive IoT applications that periodically perform inference tasks over spatially dispersed sensor data streams. We developed both an exact optimisation model and a heuristic strategy to solve the allocation problem. Numerical results demonstrated that the proposed method outperforms the benchmark policies. Interesting directions for future work include studying the generalisation of our framework to include network slicing and different revenue and utility models.

References

[1] L. Tian, H. Wang, Y. Zhou, and C. Peng, "Video big data in smart city: Background construction and optimization for surveillance video processing," *Future Generation Computer Systems*, vol. 86, pp. 1371–1382, 2018.

[2] A. Prati, C. Shan, and K. I.-K. Wang, "Sensors, vision and networks: From video surveillance to activity recognition and health monitoring," *Journal of Ambient Intelligence and Smart Environments*, vol. 11, no. 1, pp. 5–22, January 2019.

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

[4] Q. Pham and F. Fang and V. N. Ha and M. J. Piran and M. Le and L. B. Le and W. Hwang and Z. Ding, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE Access*, vol. 8, no. 116974–117017, 2020.

[5] ETSI ISG MEC, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," ETSI GS MEC 003 V2.1.1, January 2019.

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

[7] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1047–1060, 2020.

[8] C. Cicconetti, M. Conti, and A. Passarella, "Uncoordinated access to serverless computing in MEC systems for IoT," *Computer Networks*, vol. 172, p. 107184, 2020.

[9] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1359–1374, 2020.

[10] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint Computation Offloading and User Association in Multi-Task Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[11] J. Zhou, X. Zhang, and W. Wang, "Joint Resource Allocation and User Association for Heterogeneous Services in Multi-Access Edge Computing Networks," *IEEE Access*, vol. 7, pp. 12 272–12 282, 2019.

[12] W. Teng, M. Sheng, K. Guo, and Z. Qiu, "Content Placement and User Association for Delay Minimization in Small Cell Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 201–10 215, 2019.

[13] ETSI ISG MEC, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 1: System, host and platform management," ETSI GS MEC 010-1 V1.1.1, October 2017.

[14] S. Jošilo and G. Dán, "Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[15] Y. Sun, G. Feng, S. Qin, and S. Sun, "Cell Association With User Behavior Awareness in Heterogeneous Cellular Networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4589–4601, 2018.

[16] A. Costa, T. S. Ng, and L. X. Foo, "Complete mixed integer linear programming formulations for modularity density based clustering," *Discrete Optimization*, vol. 25, pp. 141–158, August 2017.

[17] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 2011.

[18] , "Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects," 3GPP, Tech. Rep. 36.814 V9.2.0 , March 2017.