IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 24, 2019, accepted April 17, 2019, date of publication April 29, 2019, date of current version May 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913564

EcoMobiFog–Design and Dynamic Optimization of a 5G Mobile-Fog-Cloud Multi-Tier Ecosystem for the Real-Time Distributed Execution of Stream Applications

ENZO BACCARELLI, MICHELE SCARPINITI, (Senior Member, IEEE), AND ALIREZA MOMENZADEH

Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00184 Rome, Italy

Corresponding author: Michele Scarpiniti (michele.scarpiniti@uniroma1.it)

This work was supported in part by the project: GAUChO – A Green Adaptive Fog Computing and networking Architecture, through the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015 under Grant 2015YPXH4W_004, and in part by the projects: Vehicular Fog: energy-efficient QoS mining and dissemination of multimedia Big Data streams (V-Fog and V-Fog2), and: SoFT: Fog of Social IoT", through the Sapienza University of Rome, Bandi 2016, 2017, and 2018.

ABSTRACT The emerging 5G paradigm will enable multi-radio smartphones to run high-rate stream applications. However, since current smartphones remain resource and battery-limited, the 5G era opens new challenges on how to actually support these applications. In principle, the service orchestration capability of the Fog and Cloud Computing paradigms could be an effective means of dynamically providing resource-augmentation to smartphones. Motivated by these considerations, the *peculiar* focus of this paper is on the *joint* and *adaptive* optimization of the resource and task allocations of mobile stream applications in 5G-supported *multi-tier* Mobile-Fog-Cloud virtualized ecosystems. The objective is the *minimization* of the computing-plus-network energy of the overall ecosystem under *hard constraints* on the minimum streaming rate and the maximum computing-plus-networking resources. To this end: 1) we model the target ecosystem energy by explicitly accounting for the virtualized and multi-core nature of the Fog/Cloud servers; 2) since the resulting problem is non-convex and involves both continuous and discrete variables, we develop an optimality-preserving decomposition into the cascade of a (continuous) resource allocation sub-problem and a (discrete) task-allocation sub-problem; and 3) we numerically solve the first sub-problem through a suitably designed set of gradient-based *adaptive* iterations, while we approach the solution of the second sub-problem by resorting to an ad-hoc-developed *elitary* Genetic algorithm. Finally, we design the main blocks of *EcoMobiFog*, a technological virtualized platform for supporting the developed solver. The extensive numerical tests confirm that the energy-delay performance of the proposed solving framework is typically within a *few per-cent* the benchmark one of the exhaustive search-based solution.

INDEX TERMS Multi-tier Mobile-Fog-Cloud ecosystems, multi-radio 5G, service models, real-time mobile stream applications, adaptive joint resource and task allocation.

I. INTRODUCTION

With smartphones becoming our symbiotic personal assistant, high-quality mobile applications are playing an important role in our life. This is mainly due to the fact that current smartphones are more and more being equipped with an increasing number of heterogeneous sensors and wireless Network Interface Cards (NICs), that make today feasible to support multimedia mobile stream applications [1]. These applications usually exploit video cameras and/or other native sensors, in order to carry out in real-time perception-based jobs, like, for example, object and/or gesture recognition and augmented-reality immersive experiences, just to cite a few. However, these applications share two main features that make them hard to be supported by current stand-alone smartphones. First, by definition, they require the continuous

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad S. Khan.

TABLE 1. Native features and synergic interplay of the pillar FC, CC and MR-5G paradigms.

FOG COMPUTING		CLOUD COMPUTING		MULTI-RADIO 5G	
Pervasive deployment	Fog servers are pervasively deployed at the network edge, in order to limit the network delay	**Centralized deployment**	Cloud datacenters sit in the backbone network and their access delays are high	**Support for multi-radio technologies**	Multiple short/long-range radios are simultaneously supported and dynamically turned ON-OFF by 5G smartphones
Light virtualization	Virtualized clones of the served smartphones are hosted by Fog servers. Container-based virtualization technologies are employed for reducing the resulting virtualization overhead	**Heavy virtualization**	Clones of the served devices are statically deployed by resorting to large-size (i.e., heavy) Virtual Machines	**Dynamic bandwidth provisioning**	Wireless bandwidth is dynamically provided to the requiring multi-radio smartphones on a per-radio basis
Support for throughput-sensitive stream applications	Fog nodes exploit low-latency short-range links for enabling fast task offloading from smartphones	**Support for delay-tolerant applications**	Being natively equipped with a large number of powerful servers, Cloud data centers may execute computing-intensive (but delay-tolerant) tasks offloaded by remote devices	**Bandwidth aggregation**	The simultaneous utilization of multiple radios allows the aggregation of the wireless bandwidth through bandwidth pooling
Energy saving	Resource-limited smartphones may save energy by leveraging proximate Fog servers as computing clones	**Energy wasting**	The access to remote Clouds by Mobile devices requires the utilization of energy-wasting multi-hop cellular links	**Ultra-low access delay**	Sub-millisecond access delays are achieved by the synergic utilization of 5G-enabled multiple radios

high-throughput processing of the data streams generated by high-data-rate sensors, in order to guarantee accuracy [1]. For example, low-resolution video streams may miss/veil object poses or human gestures and, then, may give rise to a low Quality of Service (QoS). Second, the mining and/or machine learning-based algorithms used to extract from the acquired data streams useful information are typically computation intensive. Hence, since the computing and battery capacities of current smartphones are still limited at a large extent, it could be appealing to resort to the so-called Mobile Cloud Computing (MCC) paradigm and, then, offload computation-intensive tasks to remote (i.e., distant) resource-rich Cloud data centers for their execution [2]. However, due to the delay and throughput-sensitive features of typical mobile stream applications, this solution would increase both the network traffic to be sustained by the Mobile-Cloud backhaul network and the overall service latency [2]. In principle, a more performing approach could be to allow the smartphones to leverage both their native multi-radio capability and the ultra-short latencies guaranteed by the emerging Fifth Generation (5G) network technology [3], in order to suitably allocate the offloaded application tasks over both the remote Cloud and proximate virtualized servers, generally referred to as Fog nodes [4]. An examination of Table 1 unveils why the integration of the three pillar paradigms of Fog Computing (FC), Cloud Computing (CC) and Multi-Radio 5G (MR-5G) could improve both the energy performance of smartphones and the throughput (i.e., processing rate) performance of the supported stream applications.

Fog Computing is a quite novel computing paradigm [4]. By definition, it enables pervasive *local* access to virtualized small-size pools of computing resources that can be *quickly* provisioned, *dynamically* scaled up/down and released on an on-demand basis. Proximate resource-limited mobile devices may access these resources by establishing *single-hop* communication links. The first column of Table 1 points out the native features of the FC paradigm.

Somewhat complementary features are retained by the (more traditional) Cloud Computing paradigm (see the second column of Table 1). In fact, by definition, the CC paradigm enables *ubiquitous* access to *large-size* pools of virtualized computing resources by establishing (typically) multi-hop cellular-type communication paths. Resource provisioning/releasing entails no negligible bootstrapping delays, and resource scaling embraces latencies of tens of milliseconds. Hence, offloading of computing-intensive but delay-tolerant and communication-light tasks well matches the native feature of the CC paradigm.

Thanks to its ultra-short latencies and support of multi-radio terminals, the forthcoming 5G paradigm is expected to

be an ideal ''glue'' for enabling the synergic integration of the Fog and Cloud paradigms (see the third column of Table 1). In fact, by design, 5G provides a multi-radio network platform that hosts existing 2G, 3G and 4G cellular technologies. It is envisioned that 5G may also integrate other short/long-range communication technologies (like, for example, WiFi, mobile satellite system, digital video broadcasting) by resorting to multi-tier spatial coverage based on the overlay of macro, pico, femto and other types of cells [3].

A. WHY THE CONVERGENCE OF FOG-CLOUD-5G? SOME MOTIVATING USE CASES

In order to appreciate the potential impact of the synergic integration of the three pillar paradigms of Fog Computing, Cloud Computing and Multi-Radio 5G, let us consider the general mobile operative scenario in which a user equipped with a smartphone desires to process a stream of frames of a given application. This last is composed of a number of inter-connected tasks (i.e., sub-routines, methods or threads) and it is described by the corresponding application Directed Acyclic Graph (DAG) [1]. Since the smartphone is energy limited and equipped with limited computing resources, the corresponding operative system may decide to execute each task of the current frame locally or offload it to a connected Fog or Cloud node by leveraging the 5G-enabled multi-radio capability of the smartphone.

In the sequel, we shortly review (few) emerging use cases that fit the aforementioned general scenario, so to illustrate the supporting role played by the underlying Fog-Cloud-5G integrated system (see, for example, [5] for a detailed presentation of a spectrum of Fog-supported use cases).

1) OBJECT RECOGNITION APPLICATIONS

Let us consider a mobile user who desires to quickly detect the presence/absence of a specific object from the real-time video stream captured by the camera of his/her smartphone. Since the underlying object recognition algorithm must operate on a per-frame basis, it may be too complex to be fully executed by the smartphone during an inter-frame interval. Therefore, the operative system hosted by the smartphone splits the overall algorithm into three main components, namely the object-detection, feature-extraction and object-recognition components. Afterwards, during each inter-frame interval, the first component may be executed locally by the smartphone, while the second and third ones may be executed by a proximate Fog server and a remote Cloud server, respectively. The required Mobile-Fog-Cloud data exchange is supported by the underlying WiFi and Cellular parallel connections managed by the multi-radio interfaces that equip the smartphone.

2) AUGMENTED REALITY AND IMMERSIVE MOBILE APPLICATIONS

The development of near-to-eye display technologies (like, for example, Google Glasses) is opening the doors to new types of immersive applications that exploit the so-called Augmented Reality (AR) paradigm. Just as an example, let us consider a museum, where a network of local Fog servers is strategically deployed along the visiting tour. In this scenario, beside to listen explanations through headphones, visitors may be guided in real-time through a stream of visual annotations by exploiting the WiFi connections sustained by the local Fog servers. So doing, a stream of scenes can come to life right before the visitors' eyes immersing them in ancient history. Furthermore, specific queries by the visitors may be addressed by streaming the required information from a central archive hosted by a (possibly, distant) Cloud server.

3) SMART SHOPPING CENTERS

Let us consider a multi-floor shopping center where a number of local Fog servers collectively forms an integrated multimedia database about the offered products. In this scenario, Fog servers at different floors store floor-related information that, in turn, is periodically updated by a central (possibly, remote) Cloud server. By exploiting WiFi connections, the Fog servers can collectively stream radio-navigation services to smartphone-equipped mobile users, in order to interactively guide them through the mall and annotate in real-time all visited products on their shopping lists.

The common feature of all these real-world applications is that they require the real-time stream execution of similar programs, like, for example, radio-positioning programs, object recognition programs, and 3D visual rendering programs, just to name a few. Although these programs are already available at a large extent [1], their computational complexities are typically high, so that current smartphones are not still capable of supporting their complete execution in a standing-alone way [5].

B. THE CONSIDERED MULTI-TIER MULTI-RADIO ECOSYSTEM

Motivated by this consideration, in Fig. 1, we sketch the main building blocks of the considered networked multi-tier multi-radio virtualized ecosystem for the support of task offloading from a resource-limited Mobile device. The ecosystem is composed by a Mobile device (i.e., a smartphone), a number $Q \geq 1$ of proximate Fog nodes and a remote Cloud node. A 5G-based FRAN (resp., CRAN) supports the Mobile-Fog (resp., Mobile-Cloud) wireless up/down single-hop TCP/IP connections, while a (possibly wired and/or multi-hop) Backhaul network guarantees the inter-Fog and Cloud-Fog TCP/IP connectivity.

In the considered framework of Fig. 1, the Mobile device may be equipped with multiple wireless NICs, in order to process in parallel multiple transmit-receive wireless streams. For this purpose, it is assumed that the Transport-layer of the protocol stack at the Mobile device hosts the Multi-Path TCP, i.e., MPTCP (see, for example, the contributions in [6], [7], and references therein for extensive performance analysis of MPTCP and related implementation aspects).

Virtualization is employed in 5G-supported Fog/Cloud data centers, in order to [5]: (i) dynamically multiplex the available physical computing, storage and networking

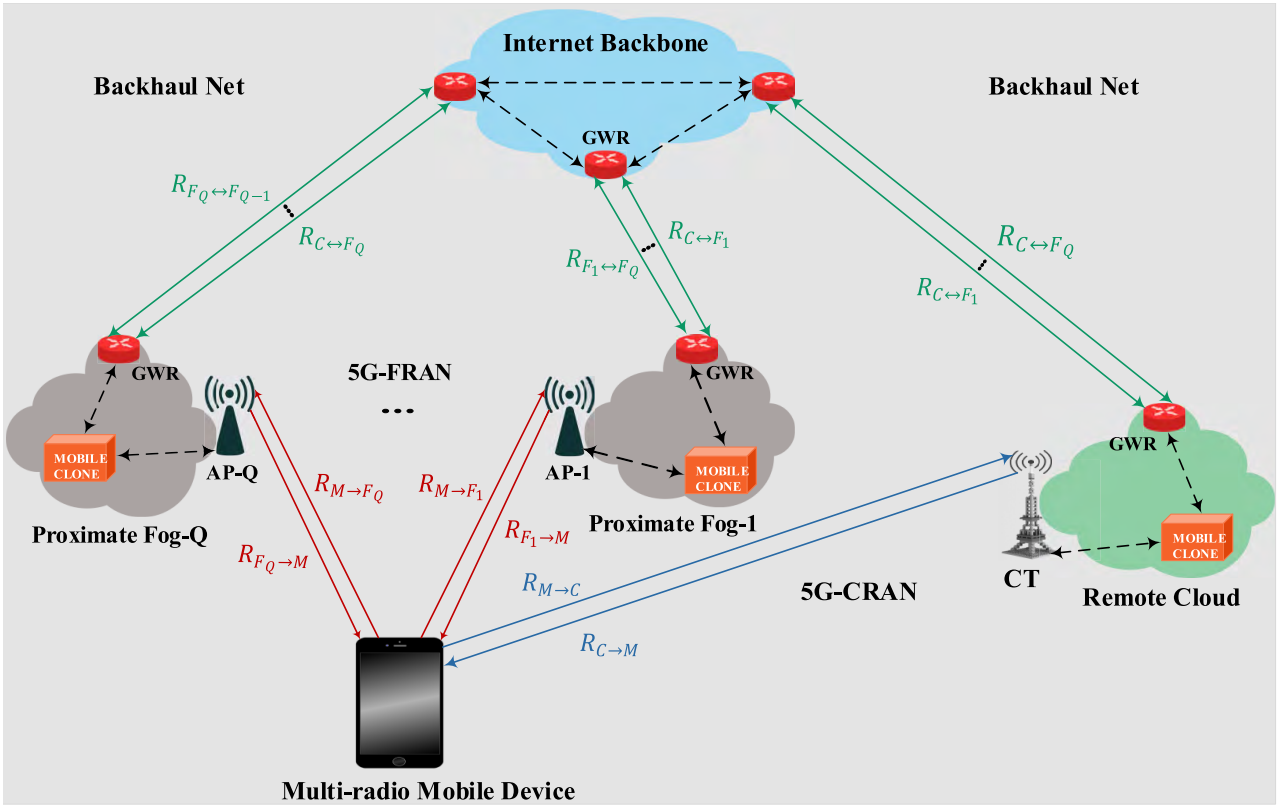

FIGURE 1. The considered 5G-supported Mobile-Fog-Cloud multi-tier ecosystem. Single (resp. double)-arrow paths indicate one-way (resp. two-way) TCP/IP connections. AP := Access Point; GWR := GateWay Router; CT := Cellular Tower; FRAN := Fog Radio Access Network; CRAN := Cloud Radio Access Network.

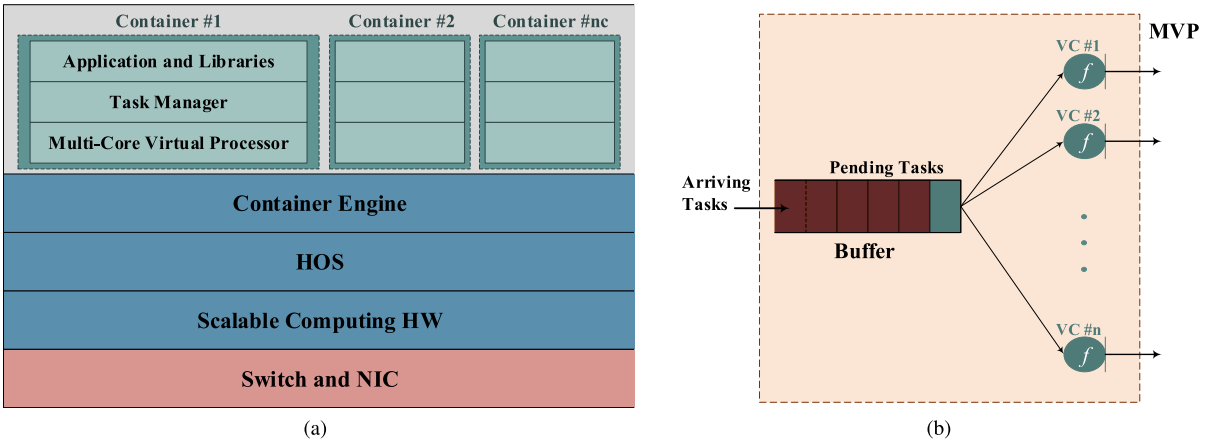

FIGURE 2. Container-based virtualization of a physical server equipping the Mobile, Fog, and Cloud nodes. (a) Virtualized server architecture; (b) Architecture of a multi-core virtual processor. HW := CPU HardWare; NIC := Network Interface Card; HOS := Host Operating System; MVP := Multi-core Virtual Processor; VC := Virtual Core; n := Number of virtual cores; f := Per-core processing frequency.

resources over the spectrum of the served mobile devices; (ii) provide homogeneous interfaces atop (possibly) heterogeneous 5G mobile devices; and, (iii) isolate the applications running atop a same physical server, so as to provide trustworthiness. Hence, according to the considered virtualized environment, the Fog and Cloud nodes of Fig. 1 are equipped with (software) clones of the Mobile device. Each clone acts as a (virtual) multi-core "server" processor and provides resource augmentation to the "client" Mobile device by processing workload on behalf of it. For this purpose, each clone is run by a container that is instantiated atop the host computing node [8]. So doing, the clone is capable of exploiting (through resource multiplexing) a slice of the physical computing and network resources of the host computing node.

Fig. 2 reports the basic elements of the container-based virtualized architecture that equips the Mobile device and each computing node of Fig. 1.

Specifically, according to Fig. 2a, each server at the Mobile, Fog and Cloud nodes hosts a number $nc \geq 1$ of containers. All the containers hosted by the same physical server share: (i) the server's Host Operating System (HOS); and, (ii) the pool of computing (i.e., CPU cycles) and networking (i.e., I/O bandwidth) physical resources done available by the CPU and NICs that equip the host server. Job of the Container Engine of Fig. 2a is to dynamically allocate to the requiring containers the bandwidth and computing resources done available by the host server. In order to execute the allocated workload on behalf of the Mobile device, each container is equipped with a Multi-core Virtual Processor (MVP). This last comprises (see Fig. 2b): (i) a buffer that stores the currently offloaded application tasks; and, (ii) a number $n \geq 1$ of (typically, homogeneous) Virtual Cores (VCs), that run at the processing frequency f dictated by the Container Engine. Therefore, goal of the Task Manager of Fig. 2a is to allocate the pending application tasks over the set of virtual cores of Fig. 2b. This is done according to the actually implemented service discipline (see Section IV).

C. MAIN CONTRIBUTIONS AND ROADMAP OF THE PAPER

On the basis of an overview of the related work carried out in Section II, we anticipate that the main contributions of our paper may be summarized as follows:

1. we carefully model both the computing and networking energy of the multi-tier ecosystem of Fig. 1 by explicitly accounting for its virtualized multi-core and multi-radio features;
2. we develop a solving approach for the delay-constrained minimization of the overall computing-plus-networking energy consumed by a stream application by performing task offloading and allocation of the per-core computing frequencies and per-connection network throughput of the ecosystem of Fig. 1 in a *joint* and *adaptive* way. Interestingly enough, the developed solving approach allows us to account for: (i) the minimum required application throughput (i.e., the minimum rate at which the stream application must be executed); (ii) the task service and scheduling disciplines actually implemented by the computing nodes of Fig. 1; (iii) the maximum allowed per-connection network throughput and per-core processing frequencies; and, (iv) the specific service model enforced by the Service Provider who manages the platform of Fig. 1. For this purpose, the proposed solving approach suitably combines gradient-based adaptive iterations with a Genetic-based elitary meta-heuristic, in order to simultaneously attain adaptive resource and task allocation;
3. we design the main building blocks and define the supported services of *EcoMobiFog*, i.e., the proposed virtualized technological platform for the actual support of the developed solving framework; and, finally,
4. we carry out extensive numerical tests for the evaluation and comparison of the energy-vs.-delay performance of the designed solving framework under a number of operative scenarios and application DAGs. In particular, (i) we compare the performance-vs.-computational complexity trade-off of the proposed solver with respect to the corresponding ones of five benchmark solvers, namely the *Only-Task Allocation*, *Only-Fog*, *Only-Mobile*, *Only-Cloud* and *Exhaustive-Search* solvers; and, (ii) we numerically test the sensitivity of the energy-delay performance of designed solver on two pillar service models, namely the *Eco-centric* and the *Mobile-centric* service models. All the reported numerical results have been carried out by the recently developed *VirtFogSim* toolbox[1].

The roadmap of the remaining part of the paper is as follows. After reviewing the related work in Section II, Section III formally introduces the main features of DAGs for mobile stream applications, while Sections IV and V are devoted to formally characterize the service/scheduling disciplines at the computing nodes and the models of the computing and network energy, respectively. Section VI introduces the afforded *Joint Optimization Problem (JOP)*, as well as its decomposition in the cascade of a *Resource Allocation Problem (RAP)* and a *Task Allocation Problem (TAP)*. Afterwards, Sections VII and VIII present the proposed solving approaches of the *RAP* and *TAP*, together with the analysis of the associated computational complexities. In Section IX, we detail the architecture of *EcoMobiFog*, i.e., the proposed technological platform for the actual support of the developed *JOP* solver. Afterwards, in Section X, we numerically test and compare the actual energy-vs.-delay performance of the proposed solving framework under a number of application scenarios and benchmark DAGs. Conclusive Section XI recaps the main results of our work and provides some hints for future research. Appendix A reports the main taxonomy of the paper, together with the meanings/roles of the main used symbols/parameters, their measuring units and simulated values. Final Appendixes B, C, D, and E present the analytical proofs of the main formal results of the paper.

Regarding the adopted notation, we point out that the arrowed subscript: $\vec{x}$ indicates a row vector, $|\mathcal{V}|$ is the size (i.e., the cardinality) of the set $\mathcal{V}$, $\lceil . \rceil$ (resp., $\lfloor . \rfloor$) is the ceil (resp. floor) function, while $[M] \stackrel{\text{def}}{=} \left[m_{i,j}\right]_{i,j=1}^{L}$ denotes an $(L \times L)$ matrix, whose (i,j)-th element is $m_{i,j}$. Furthermore, the symbol $u_{-1}(y)$ indicates the unit-step Heaviside function (i.e., $u_{-1}(y) \stackrel{\text{def}}{=} 0$ for $y \leq 0$, and $u_{-1}(y) \stackrel{\text{def}}{=} 1$, otherwise), while $\delta(y)$ is the Kronecker's delta function (i.e., $\delta(y) \stackrel{\text{def}}{=} 1$ for $y = 0$, and $\delta(y) \stackrel{\text{def}}{=} 0$, otherwise)

Finally, formal assumptions are marked by bullets.

II. RELATED WORK

An overview of the large body of literature related to the broad topic of MCC points out that Mobile Edge Computing (MEC) is another computing model that is sometimes (mis)understood as a synonymous of Fog Computing [9]–[11]. In this paper, we distinguish these two paradigms. The main reason is that, in the MEC paradigm, proximate

[1] Available online at: https://github.com/mscarpiniti/VirtFogSim

network nodes are exploited for *only* providing resource augmentation to Mobile devices by exploiting single-hop connections. As a consequence, the resulting MEC computing infrastructure is inherently composed by *only* two tiers of entities, i.e., the "client" Mobile devices and the "server" edge nodes. In contrast, Fog Computing aims to harness computing across the *full path* followed by the data to be processed, and this path may include *multiple* (possibly, hierarchically organized) tiers of intermediate server nodes, as well as a remote Cloud data center (see Fig. 1). Therefore, Fog infrastructures are natively composed of three or more tiers of nodes. So doing, the computational needs of mobile devices and edge nodes can be supported by cloud-like proximate resources or, alternatively, processed data can be transported from the remote cloud node to the edge of the network [11].

According to this observation, we note that a first (rich) branch of research work on task placement considers two-tier MEC scenarios that involve only two computing nodes, i.e., a first node hosted by the Cloud or a proximate MEC data center, and a second node running atop the mobile device (see the recent tutorial on MEC in [12]). A second (substantial) research branch focuses on the so-called task migration/allocation problem, where two physical computing nodes (like, for example, a Mobile device and a cloud or MEC node) are still involved [13]–[15].

However, to date, less considerable work seems to be available on the core problem tackled by this paper, i.e., the dynamically optimized placement of application tasks over *three* or *higher order-tier* networked computing platforms. In this regard, an overview of this last set of work shows that, roughly speaking, the related on-going research is moving along three main lines, namely: (i) the optimized placement of multi-task applications in multi-tier data-centers; (ii) the design of multi-tier computing architectures for MCC and related management protocols; and, (iii) the design of task offloading and resource allocation algorithms for multi-tier mobile computing environments.

A first group of contributions in [16]–[20] focuses on the optimized placement of multi-task applications in multi-tier data-centers. In this regard, the authors of [16] develop an algorithm for the minimization of the total cost of task placement under load balancing constraint. The proposed algorithm is based on Linear Program (LP) relaxation, its computational complexity scales linearly with the number of tasks to be allocated and does not allow resource sharing of the computational resources of the underlying physical nodes. In order to address this last point, the authors of [17] propose an algorithm for mapping application DAGs with tree topologies onto the physical graphs of networked computing nodes. The goal is still the minimization of the total cost of the performed mapping under constraints on the maximum utilization of each link of the underlying physical graph. Being the afforded problem NP-hard, a suboptimal low-complexity online version is also developed in [17] by relying on a suitable linear relaxation of the afforded problem. The paper in [18] proposes an LP-based algorithm for the (dual) problem of the offline mapping of DAG paths onto data centers with tree topology. The goal is the minimization of link congestion so that load balancing of the computing nodes is not included by the adopted objective function. Furthermore, the constraints considered in [18] force the DAG tasks to be only mapped into the leaves of the tree-shaped graph of the underlying data center. The contributions in [19] and [20] focus on the (quite recent) problem of the embedding of the service chains. Triggered by the emerging trend of Network Function Virtualization (NFV), the common goal of these contributions is to map a linear application DAG (i.e., a DAG with chain topology) onto the physical path joining fixed source and sink computing nodes, so that a sequential chain of operations may be performed on the data packets moving from the source node to the destination one. However, the topic of link placement optimization is not considered by these papers. Overall, like our contribution, these first set of papers consider the general problem of optimized task placement onto multi-tier networked computing infrastructures. Nevertheless, *unlike* our contribution, their solutions are not adaptive.

A second group of contributions in [21]–[27] tackles with the design aspects and architectures of multi-tier computing technological platforms for the support of MCC applications. For this purpose, the authors of [21] propose a code offloading framework, i.e., *MAUI*, that supports method-level energy-aware offloading for mobile applications described by DAGs. The developed framework allows to annotate methods and retrieves information from a set of profilers, in order to take decisions on whether to offload. The remarkable feature shared by the *Thinkair*, *Cloudlet* and *Music* frameworks in [22]–[24] is that they rely on the virtualization of the served mobile devices, in order to enable them to offload computing-intensive tasks to their clones running on distant nodes. As our contribution, all these frameworks consider virtualized multi-tier offloading technological platforms. However, *unlike* our framework, all these papers subsume stable (i.e., static) operative environments, which may be over-optimistic under failure-prone networking scenarios. Later, a number of works proposes to consider other types of resources for offloading. For example, the authors of [25] develop an architecture composed by wearable devices, mobile devices and cloud for code offloading. As our contribution, the goal of [25] is to allow the execution of computing-intensive applications on wearable devices through task offloading towards proximate/remote server nodes. However, *unlike* our contribution, in [25], the impact on the offloading performance of the (possibly, time-varying) feature of the underlying wireless connections is not considered. The focus of [26] is on the design of a system architecture (i.e., *StreamCloud*) that supports fine-grained offloading of tasks of stream applications from a mobile device towards distant serving nodes. In order to solve the underlying decision process, this paper presents a Genetic-based meta-heuristic, that is capable to maximize the application throughput under the constraint on the available maximum wireless bandwidths. Hence, like our work,

also this contribution considers the application throughput as a pivotal performance metric for stream applications and resorts to the Genetic paradigm as solving approach. However, *unlike* our work, the paper in [26]: (i) does not perform dynamic optimization of the network and/or computing resources; and, (ii) does not consider the network and/or computing energy consumption as target objectives to be minimized. These aspects are, indeed, addressed at some extent by the so-called *mCloud* framework recently proposed in [27]. Specifically, the authors of this last contribution develop a technological platform for task offloading from a mobile device to remote Clouds and/or nearby Fog nodes. The target is the minimization of the task execution times by leveraging context-awareness, in order to dynamically select the most energy-saving wireless connection over the ones simultaneously managed by the Mobile device. Hence, like our work, the resulting offloading framework of [27] accounts for: (i) the presence of a multi-tier networked computing infrastructure, that is capable to provide resource augmentation to resource-limited mobile devices; and, (ii) the time-varying and heterogeneous power-vs.-delay profiles of the wireless connections managed by the mobile device. However, *unlike* our contribution, the *mCloud* framework: (i) does not perform dynamic scaling of the computing resources available at the Mobile and Cloud/Fog nodes; (ii) the tasks to be offloaded are considered mutually independent, i.e., no precedence constraints are assumed to be enforced by the underlying application DAG; (iii) the impact of the service discipline at the computing nodes is not modeled; and, (iv) all processing nodes are assumed single-core.

A third set of contributions in [28]–[35] affords the (broad) topic of the optimized design and performance evaluation of task offloading and resource allocation algorithms for mobile multi-tier computing environments. In this regard, the authors of [28] develop a semi Markovian-based framework for triggering the offloading decision, that aims at attaining a good trade-off between the contrasting requirements of low DAG execution times and low energy consumption. However, the developed decision framework assumes a stable network condition, that is quite over-optimistic in mobile environments. In order to reduce the decision-delays that inherently affect the solving approaches based on the Markov Decision Process, the contribution in [29] resorts to profile and cache the already computed offloading planes, while [30] proposes a proactive approach that exploits location-awareness for performing mobility prediction. The common feature shared by the contributions in [31]–[35] is that they model the underlying application as a weighted DAG, in order to account for the task dependencies and related computing/communication workloads. Specifically, [31] pursues the criterion of workload balancing between mobile device and distant servers, in order to develop a heuristic DAG-partitioning algorithm for the reduction of the resulting execution time. The paper in [32] investigates the latency of DAG executions under constraints on the available computing/communication resources, and develops a polynomial-complexity approximate solution with guaranteed performance. The goal of [33] is the minimization of the energy consumed by the mobile device through task offloading. For this purpose, both the scheduling and offloading decisions are jointly optimized by numerically solving a suitable integer program. In order to efficiently cope with the fading phenomena impairing the mobile channels supporting task offloading, the paper in [34] performs the delay-constrained minimization of the average energy consumption of the mobile device. To this end, in [34], the afforded problem is turned into a stochastic shortest path problem, that is solved through suitable *one-climb* offloading policies. Finally, the recent contribution in [35] develops online approximate algorithms with poly-log competitive ratios for the load-balanced mapping of an application DAG onto a networked computing graph under constraints on the link utilization.

In summary, on the basis of the carried out research overview, we may conclude that the peculiar feature of our contribution is as follows. It aims at maximizing the energy efficiency of task offloading of *throughput-constrained* mobile stream applications. To this end, a *dynamic* framework that leverages the *virtualization* of the available computing/networking resources by *jointly* optimizing resource and task allocation. This is done by accounting for: (i) an ecosystem of (possibly, heterogeneous) offloading destinations, that inter-communicate through (possibly, heterogeneous) TCP/IP 5G connections; (ii) the dynamically changing network conditions; and, (iii) the service policy actually enforced by the involved Service Providers.

III. MODELING STREAM APPLICATIONS

Real-life applications are composed by a number of basic tasks that can exhibit arbitrary sets of inter-dependencies. In general, a suitable description of these last may be exploited by the mobile device of Fig. 1, in order to improve the energy performance of the carried out task-offloading process.

For this purpose, application Component Dependency Graphs (also referred to as application Task-Call Graphs) may be utilized [1]. From a formal point of view, an application component dependency graph is a DAG: $\mathcal{G}_{APP} \stackrel{\text{def}}{=} (\mathcal{V}, E)$, whose node set: $\mathcal{V} \stackrel{\text{def}}{=} \left\{i : i = 1, \ldots, V \stackrel{\text{def}}{=} |\mathcal{V}|\right\}$ represents the application tasks, while the set: $E \stackrel{\text{def}}{=} \left\{e_{ij}, i \in \mathcal{V}, j \in \mathcal{V}\right\}$ of the directed edge captures the inter-task dependencies [1]. Being a weighted graph, an application DAG is formally characterized by [1]:

i. the binary-valued matrix: $[A] \stackrel{\text{def}}{=} \left[a_{ij}\right]_{i,j=1}^{V}$ of the pairwise task adjacencies;
ii. the real-valued matrix: $[D_A] \stackrel{\text{def}}{=} \left[d_{ij}\right]_{i,j=1}^{V}$ of the edge weights, with d_{ij} being the weight (measured in (bit)) of the (i, j)-th edge $e_{ij} \in E$; and,
iii. the vector: $\vec{s} \stackrel{\text{def}}{=} [s_1, \ldots, s_V]$ of the task sizes, with s_i (measured in (bit)) being the workload to be sustained for the execution of the i-th task of the DAG.

TABLE 2. Profiled processing densities of some classes of real-life mobile stream applications [15].

Application class	**Application density** (CPU cycle/bit)
Video transcoding	$200 - 1200$
Processing of 400 frames of video game	$2400 - 1200$
Gesture recognition from high-resolution photos	$2500 - 32000$
Virus scanning	$33000 - 37000$

Furthermore, according to, for instance, [12], [13], in the sequel, we assume that every application DAG describing a mobile application retains the following six defining properties:

- each node $i \in \mathcal{V}$, with $i \neq 1$ and $i \neq V$, has an in-degree and an out-degree of (at least) one. The in-degree of the first node vanishes, while the out-degree of the last node is zero;
- each intermediate node $i \in \mathcal{V}$, with $i \neq 1$ and $i \neq V$, has at least one directed path from the first task and at least one directed path to the last task, so that the first and last tasks are the root and the sink of the considered application DAG, respectively;
- the DAG is loop-free, so that each root-to-sink directed path is of finite length;
- each task must be processed by one and only one computing node of the ecosystem of Fig. 1;
- both the first and last nodes cannot be offloaded, and, then, must be executed by the Mobile device;
- a clone of the Mobile device is already deployed at the Cloud node and at each Fog node of Fig. 1. Each clone has the same software stack as its associated Mobile device and is equipped with the application DAG to be executed.

Regarding the rationales behind the above assumptions, five main explicative remarks are in order.

First, the first three assumptions are (at least) necessary, in order to have finite DAG execution times.

Second, depending on the more or less fine granularity of the considered DAG, in our framework, a task may represent a routine, a method or a thread. Hence, the fourth assumption is compliant with the atomic nature of these entities.

Third, the fifth assumption reflects the fact that the executions of first and last tasks of real-life mobile applications typically require the utilization of input/output *hardware* cards (like, screens, keyboards, sensors, photo/video-cameras, microphones and similar) that are hosted by the Mobile device, so that these tasks are not off-loadable [1], [12].

Fourth, in practice, the sixth assumption may be actuated by performing suitable real-time migrations of the underlying containers. Since this specific topic has been recently addressed, for example, in [7], we limit to note that, under this last assumption, the Mobile device only needs to transmit a small volume of signaling data to the mobile clones, in order to perform synchronization and indicate the tasks to be remotely executed. Therefore, in the sequel, we do not consider the time and energy consumption induced by the transmission of signaling data.

Finally, according to a network-oriented point of view, in our framework, the tasks sizes are assumed to be expressed in bit. In practical, the CPU workload: $\hat{s}_i$ (measured in (CPU cycle)) required by the execution of the i-th task is related to the corresponding task size: s_i (bit) through the following product formula [15]:

$$\hat{s}_i = pd \times s_i. \tag{1}$$

In the above relationship, pd (expressed in (CPU cycle/bit)) is the so-called processing density of the considered application. Since it measures the average number of CPU cycles that are required for the processing of a single application bit, its actual value depends on the computing load of the underlying application. For illustrative purposes, Table 2 reports the values of the processing densities of some benchmark mobile stream applications.

IV. THE CONSIDERED SERVICE AND SCHEDULING DISCIPLINES

The goal of this section is to introduce the basic formal notation and operative assumptions about the networked ecosystem of Fig. 1. In this regard, after indicating by:

$$\mathcal{A} \stackrel{\text{def}}{=} \{M, F_1, \ldots, F_Q, C\}, \tag{2}$$

and

$$\mathcal{BHS} \stackrel{\text{def}}{=} \{F_1, \ldots, F_Q, C\} \equiv \mathcal{A} \setminus \{M\}, \tag{3}$$

the set of the available computing nodes and that of the corresponding backhaul segment respectively (see Fig. 1), let:

$$R_{N_1 \to N_2} \text{ (bit/s)}, \quad N_1 \neq N_2, \text{ with } N_1, N_2 \in \mathcal{A}, \tag{4}$$

and

$$R_{N_1 \leftrightarrow N_2} \text{ (bit/s)}, \quad N_1 \neq N_2, \text{ with } N_1, N_2 \in \mathcal{A}, \tag{5}$$

indicate the throughput (i.e., the transport rate) of the one-way TCP/IP transport connection from node N_1 to node N_2, and that of the corresponding two-way symmetric full-duplex connection. Furthermore, since each task must be processed by only one computing node, let:

$$\vec{x} = [x_1, \ldots, x_V], \tag{6}$$

be the V-dimensional task allocation (row) vector, whose i-th discrete-valued scalar component:

$$x_i \in \mathcal{A}, \quad i = 1, \ldots, V, \tag{7}$$

indicates the computing node that must process the i-th task of the assigned application DAG.

A. SERVICE DISCIPLINES AT THE COMPUTING NODES AND RELATED SERVICE TIMES

According to the virtualized node architecture reported in Fig. 2, let: $f_{i,N}$ (bit/s), $i = 1, \ldots, V$, $N \in \mathcal{A}$ be the processing frequency that the Container Engine of Fig. 2 allocates for the execution of the i-th task of size s_i (bit). Hence, by definition, the resulting service time: $T_{i,N}^{(SER)}$ (s) measures the processing time of the i-th task, and, then, it is formally defined as follows [36]:

$$T_{i,N}^{(SER)} \stackrel{\text{def}}{=} s_i / f_{i,N}, \quad i = 1, \ldots, V; \; N \in \mathcal{A}. \tag{8}$$

Since the form assumed by $f_{i,N}$ depends on the specific service discipline adopted by node N for the processing of the assigned tasks, in the sequel, we limit to assume that [36]:

- $f_{i,N}$ is proportional to the total computing capacity $n_N f_N$ (bit/s) available at node N (see Fig. 2).

In this regard, we note that two examples of service disciplines of practical relevance that meet the above assumption are the *SEQuential (SEQ)* service discipline and the *Weighted Processors Sharing (WPS)* one (see, for example, [36, Chapter 4]).

By design, we shortly note that, under the SEQ service discipline, each task is *individually* processed according to a specified sequential ordering. Hence, $f_{i,N}$ equates, by design, the full per-node processing capability (that is, $f_{i,N} \equiv n_N f_N$), and, then, (see Eq. (8))

$$T_{i,N}^{(SER;SEQ)} \stackrel{\text{def}}{=} s_i / (n_N f_N), \quad i = 1, \ldots, V. \tag{9}$$

Under the WPS service discipline, the tasks assigned to the computing node N are processed in parallel by following a weighted round-robin task scheduling [36]. Specifically, the computing frequency at which the i-th task is processed equates to:

$$f_{i,N} \equiv \left(\frac{\phi_i}{\sum_{j=1}^{V} \phi_j \delta(x_j - N)} \right) \times n_N f_N, \tag{10}$$

for $i = 1, \ldots, V$, where [36]: (i) the (dimensionless and positive) weight coefficient ϕ_i ($i = 1, \ldots, V$), fixes the relative priority level of the i-th task; and, (ii) the delta-terms at the denominator of the above equation assure that the processing capability $n_N f_N$ available at node N is shared only by the tasks that are actually to be processed by node N.

Per-Node Total Service Times: The resulting total service time $T_N^{(SER)}$ (s) at node $N \in \mathcal{A}$ is defined as the total time spent by the node for processing *all* the assigned tasks, under the assumption that all the data needed for the task processing is *already available* at node N (i.e., by definition, $T_N^{(SER)}$ does *not* account for the inter-node network-induced transport delays [36]).

Since also the analytical expression of $T_N^{(SER)}$ heavily depends on the adopted service discipline, in the sequel, we limit to assume that [36]:

- $T_N^{(SER)}$ is proportional to: $1/(n_N f_N)$; and,
- $T_N^{(SER)}$ does not decrease when at least one size of the assigned tasks increases.

Just as illustrative examples, the expression assumed by the cumulative service time under the (aforementioned) *SEQ* service discipline is sum-like, i.e.,

$$\begin{aligned} T_N^{(SER;SEQ)} &= \sum_{i=1}^{V} T_{i,N}^{(SER;SEQ)} \, \delta(x_i - N) \\ &\equiv \frac{1}{n_N f_N} \times \left(\sum_{i=1}^{V} s_i \, \delta(x_i - N) \right), \end{aligned} \tag{11}$$

while the following max-type formula holds for the *WPS* case:

$$\begin{aligned} T_N^{(SER;WPS)} &= \max_{1 \le i \le V} \left\{ T_{i,N}^{(SER;WPS)} \, \delta(x_i - N) \right\} \\ &\equiv \frac{1}{n_N f_N} \times \max_{1 \le i \le V} \left\{ \frac{s_i \, \delta(x_i - N)}{\left(\frac{\phi_i}{\sum_{j=1}^{V} \phi_j \, \delta(x_j - N)} \right)} \right\}. \end{aligned} \tag{12}$$

B. PER-TASK EXECUTION TIMES

The impact of the network-induced delays on the performance of the ecosystem of Fig. 1 is accounted for by the corresponding execution time $T_{i,N}^{(EXE)}$ (s) of the i-th task at node $N \in \mathcal{A}$. It is formally defined as the summation [36]:

$$T_{i,N}^{(EXE)} \stackrel{\text{def}}{=} T_{i,N}^{(SER)} + T_{i,N}^{(NET)}, \quad i = 1, \ldots, V; \; N \in \mathcal{A}, \tag{13}$$

of the (already introduced) service time $T_{i,N}^{(SER)}$ and the network time $T_{i,N}^{(NET)}$, needed for the transport to node N of *all* data that is required by the execution of the i-th task. Hence, directly from the (previously reported) definition of the throughput of the one-way connections, the following relationship holds:

$$T_{i,N}^{(NET)} = \sum\nolimits_{\substack{N_1 \in \mathcal{A} \\ N_1 \neq N}} \left(\frac{vl_i^{(N_1 \to N)}}{R_{N_1 \to N}} \right), \quad i = 1, \ldots, V. \tag{14}$$

In the above equation, $vl_i^{(N_1 \to N)}$ (bit) indicates the volume of data that must be transported from node N_1 to node N for the execution of the i-th task at node N. Hence, from the definitions of the adjacency and edge weight matrices of the considered application DAG, the following relationship holds for the computation of $vl_i^{(N_1 \to N)}$, $i = 1, \ldots, V$:

$$vl_i^{(N_1 \to N)} \stackrel{\text{def}}{=} \left(1 + \overline{NF}_{N_1 \to N}\right) \times \sum_{j=1}^{V} a_{ji} d_{ji} \delta(x_j - N_1), \tag{15}$$

where $N_1 \neq N$, and $N_1, N \in \mathcal{A}$.

Before proceeding, three main explicative remarks about the relationships in (14) and (15) are in order.

First, since the (non-negative and dimensionless) term $\overline{NF}_{N_1 \to N}$ is the average failure rate of the one-way connection: $N_1 \to N$, the first factor present in (15) is the average overhead in the volume of the transported data that is induced by connection-failure phenomena.

Second, the sum-form of the expression of $T_{i,N}^{(NET)}$ in (14) applies when the multiple data streams arriving at the receiving node N are processed in a sequential way, so that their network delays are added up. However, in our framework, the mobile device is equipped by multiple NICs that, in principle, could operate in parallel. In this case, the sum-type expression in (14) should be replaced by the following max-type one [36]:

$$T_{i,N}^{(NET)} = \max_{\substack{N_1 \in \mathcal{A} \\ N_1 \neq N}} \left\{ \frac{vl_i^{(N_1 \to N)}}{R_{N_1 \to N}} \right\}. \tag{16}$$

Since the point-wise maximum of convex function is still a convex function [37], we anticipate that the convex/nonconvex nature of the constrained optimization problem to be afforded remains unchanged under both cases. Furthermore, the parallel processing of multiple received streams may induce out-of-order phenomena, that, in turn, introduce additional queue delays at the Transport Layer of the receiving nodes [7]. Hence, on the basis of these considerations, without substantial loss of generality, in the sequel, we assume that $T_{i,N}^{(NET)}$ is given by the sum-type expression in (14).

Finally we note that, since each virtual processor is equipped with a buffer that is reserved for the temporary storage of the assigned tasks (see Fig. 2), the definition in (13) of the per-task execution time $T_{i,N}^{(EXE)}$ automatically accounts for the queue delay at the underlying computing node N [36].

C. PER-DAG EXECUTION TIMES AND INTER-NODE TASK SCHEDULING DISCIPLINES

By definition, the DAG execution time T_{DAG} (s) is the time interval between the instant at which begins the execution of the first task and the instant at which ends the execution of the last task. From a formal point of view, T_{DAG} is a function [36]:

$$T_{DAG} = \mathcal{X}\left(\left\{ T_{i,N}^{(EXE)} \delta\left(x_i - N\right), i = 1, \ldots, V; N \in \mathcal{A} \right\} \right), \tag{17}$$

of both the set of the (previously modeled) per-task execution times and task allocation vector $\vec{x}$ in (6). Furthermore, the specific form of the $\mathcal{X}(.)$ function depends on the actually adopted inter-node Task Scheduling Discipline (TSD). By definition, it dictates the (statically or dynamically configured) ordering in which the computing nodes process the sets of the assigned tasks [36]. Since a number of TSDs have been even recently considered in the open literature [33], in the sequel, we assume the adopted TSD to be assigned and, then, we limit to point out two general assumptions on the resulting T_{DAG} that are typically guaranteed by the TSDs of practical interest. Specifically, we assume that [36]:

- T_{DAG} is a non-decreasing function of each per-task execution time $T_{i,N}^{(EXE)}$, $i = 1, \ldots, V, N \in \mathcal{A}$; and,
- T_{DAG} is a jointly convex function of the per-task execution times $\left\{ T_{i,N}^{(EXE)},\ i = 1, \ldots, V,\ N \in \mathcal{A} \right\}$.

Just as practical examples of TDSs that meet the above assumptions, we shortly address the *Sequential Task Scheduling* (STS) and the *Parallel Task Scheduling* (PTS) disciplines [36].

By definition, the STS discipline forces the nodes to perform their computation in a *sequential* way. Hence, this discipline does *not* allow inter-node parallel task executions and, then, it is applied when the sets of tasks assigned to the various computing nodes cannot be processed in parallel. As a matter of fact, the corresponding DAG execution time assumes the following sum-type expression:

$$T_{DAG}^{(STS)} = \sum_{i=1}^{V} \sum_{N \in \mathcal{A}} T_{i,N}^{(EXE)} \delta\left(x_i - N\right), \tag{18}$$

where the delta terms in the inner summation guarantee that the i-th task is executed by a single computing node.

By definition, the PTS discipline forces the sets of tasks assigned to the computing nodes to be processed in *parallel*, so that it may be applied when *no* interdependence is present. As a consequence, the resulting DAG execution time is given by the following max-type expression [36]:

$$T_{DAG}^{(PTS)} = \max_{1 \leq i \leq V} \left\{ \sum_{N \in \mathcal{A}} T_{i,N}^{(EXE)} \delta\left(x_i - N\right) \right\}. \tag{19}$$

By direct inspection, it can be viewed that both the above expressions meet the previously reported general assumptions on T_{DAG}.

V. MODELING THE COMPUTING AND NETWORKING ENERGY CONSUMPTION

The goal of this section is threefold. First, we introduce the cost parameters used to formally feature the Software-as-a-Service (SaaS) policy enforced by the Cloud and Fog Service Providers that manage the ecosystem of Fig. 1. Second, we develop the formal models to profile the power and energy consumption of the virtualized multi-core processors that equip the device clones at the computing nodes (see Fig. 2). Third, we pass to model the companion power and energy models of the TCP/IP-based 5G network connections that support the FRAN, CRAN and Backhaul segments of the overall network infrastructure of Fig. 1. Interestingly enough, we anticipate that the models developed for both the computing and network power/energy explicitly account for the specific SaaS server policy applied by the Service Providers.

A. SaaS POLICIES FOR VIRTUALIZED MOBILE-FOG-CLOUD NETWORKED ECOSYSTEMS

In this regard, we note that, since the ecosystem of Fig. 1 relies on container and 5G-based virtualization technologies for multiplexing of the underlying physical computing and networking resources, each device clone runs atop an isolated environment and this makes the SaaS model be applicable [38]. According to this service model, the user who manages the mobile device may be charged on the basis of the networking and computing resources that are actually wasted by the corresponding device clones of Fig. 1. Specifically, since the final goal of the ecosystem of Fig. 1 is to save energy by suitably pricing it, the Cloud and Service Providers may enforce one of the following three SaaS-based policies [38]:

i. the virtualized computing/networking resources utilized by the device clones at the proximate Fog nodes are priced or they are for free;
ii. the virtualized computing/networking resources wasted by the device clone running at the remote Cloud node are subject to "flat" pricing policies or they are metered and priced on a per-usage basis; and,
iii. the user is whether or not interested to minimize the energy consumption of his/her own mobile device.

Since, in our framework, Fog Providers, Cloud Providers and mobile users act as independent actors, any combination of the aforementioned three basic pricing policies could be applied. We anticipate that, in order to account for this consideration, we introduce three binary-valued parameters, i.e.:

$$\theta_M \in \{0,1\}, \quad \theta_F \in \{0,1\}, \quad \theta_C \in \{0,1\}, \tag{20}$$

whose settings are dictated by the SaaS policies actually implemented by the three mentioned actors (see Remark 2 of Section VI-A for the formal description of the role played by the theta parameters in (20)).

B. MODELING THE COMPUTING ENERGY IN MULTI-CORE VIRTUALIZED EXECUTION ENVIRONMENTS

According to a number of (even recent) contributions (see, for example, [39] and references therein), the computing energy $\mathcal{E}_N$ (Joule) wasted by the device clone at node $N \in \mathcal{A}$ for the execution of the assigned tasks is the summation of a static $\mathcal{E}_N^{(STA)}$ (Joule) part and a dynamic $\mathcal{E}_N^{(DYN)}$ (Joule) part.

Specifically, the static part accounts for the energy wasted by the clone in the idle state (i.e., the clone is turned ON but it is not running). As a consequence, when the clone is actually turned ON for DAG execution, $\mathcal{E}_N^{(STA)}$ equates the following product: $\mathcal{E}_N^{(STA)} \stackrel{\text{def}}{=} \mathcal{P}_N^{(STA)} \times T_{DAG}$, where T_{DAG} is the (previously introduced) DAG execution time, and $\mathcal{P}_N^{(STA)}$ (Watt) is the corresponding per-clone computing static power. After considering that each physical server at node N consumes: $\mathcal{P}_{CPU-N}^{(IDLE)}$ (Watt) unit of power for sustaining $nc_N \geq 1$ containers in the idle state, (see Fig. 2), the per-clone computing static power may be, in turn, modeled as [39]: $\mathcal{P}_N^{(STA)} = \mathcal{P}_{CPU-N}^{(IDLE)}/nc_N, N \in \mathcal{A}$.

Passing to model the dynamic part of the computing energy, we note that, by definition, a clone processes the assigned tasks over a time interval equal to the (previously defined) service time. Therefore, the dynamic component: $\mathcal{E}_N^{(DYN)}$ of the per-clone computing energy equates the product: $\mathcal{E}_N^{(DYN)} \stackrel{\text{def}}{=} \mathcal{P}_N^{(DYN)} \times T_N^{(SER)}$, where $\mathcal{P}_N^{(DYN)}$ (Watt) is the dynamic power consumed by the clone for computing purpose. By definition, this last power depends on three main factors, namely [39]:

1) the number of cores n_N equipping the virtual processor at node $N \in \mathcal{A}$ (see Fig. 2);
2) the corresponding computing frequency f_N (bit/s) at which the Container Engine of Fig. 2 does run the available cores; and,
3) the fraction $r_N \in [0,1]$ of the overall dynamic power: $\mathcal{P}_N^{(DYN)}$ that is shared by all cores for common operation.

Hence, according to, for example, [39], $\mathcal{P}_N^{(DYN)}$ may be formally profiled through the following power-like expression: $\mathcal{P}_N^{(DYN)} = n_N (1-r_N) k_N (f_N)^{\gamma_N}$, where: (i) γ_N is a dimensionless shaping exponent (i.e., typically $\gamma_N \geq 2$); and, (ii) the positive constant k_N (Watt/(bit/s)$^{\gamma_N}$) accounts for the common power profiles of the utilized (homogeneous) cores [39].

Overall, on the basis of the above considerations, the computing energy wasted by the mobile clone at node $N \in \mathcal{A}$ may be analytically profiled as follows:

$$\mathcal{E}_N = \underbrace{\overbrace{\left(\mathcal{P}_{CPU-N}^{(IDLE)}/nc_N\right)}^{\mathcal{P}_N^{(STA)}} \times T_{DAG} \times u_{-1}\left(\sum_{i=1}^{V} \delta(x_i - N)\right)}_{\mathcal{E}_N^{(STA)}} + \underbrace{\overbrace{n_N (1-r_N) k_N (f_N)^{\gamma_N}}^{\mathcal{P}_N^{(DYN)}} \times T_N^{(SER)}}_{\mathcal{E}_N^{(DYN)}}, \tag{21}$$

where the unit step-size factor: $u_{-1}\left(\sum_{i=1}^{V} \delta(x_i - N)\right)$ accounts for the fact that the device clone at node N is actually turned ON only when it must process at least a pending task.

C. MODELING THE NETWORKING ENERGY OF THE INTER-NODE CONNECTIONS

Let $\mathcal{E}_{N_1 \leftrightarrow N_2}$ (Joule) be the network energy consumed by the two-way (i.e., bi-directional) end-to-end TCP/IP Transport-layer connection between the computing nodes N_1 and N_2, with $N_1 \neq N_2$, and $N_1, N_2 \in \mathcal{A}$. Since it equates to the summation:

$$\mathcal{E}_{N_1 \leftrightarrow N_2} = \mathcal{E}_{N_1 \rightarrow N_2} + \mathcal{E}_{N_2 \rightarrow N_1}, \tag{22}$$

of the corresponding energy $\mathcal{E}_{N_1 \rightarrow N_2}$ (Joule) and $\mathcal{E}_{N_2 \rightarrow N_1}$ of the underlying one-way (i.e., directed) Transport connections: $N_1 \rightarrow N_2$ and $N_2 \rightarrow N_1$ respectively, we may directly focus on the modeling of $\mathcal{E}_{N_1 \rightarrow N_2}$. According to the analytical and experimental models reported, for example, in [40],

it may be profiled as the summation of a static part and dynamic one as in:

$$\mathcal{E}_{N_1\to N_2} = \mathcal{E}^{(STA)}_{N_1\to N_2} + \mathcal{E}^{(DYN)}_{N_1\to N_2}. \tag{23}$$

The static component: $\mathcal{E}^{(STA)}_{N_1\to N_2}$(Joule) may be expressed, in turn, as in:

$$\mathcal{E}^{(STA)}_{N_1\to N_2} = \mathcal{P}^{(STA)}_{N_1\to N_2} \times T_{DAG} \times u_{-1}\left(\sum_{i=1}^{V}\sum_{j=1}^{V} a_{ij}\delta\left(x_i - N_1\right)\delta\left(x_j - N_2\right)\right), \tag{24}$$

with

$$\mathcal{P}^{(STA)}_{N_1\to N_2} = \theta_{N_1}\mathcal{P}^{(IDLE)}_{NET-N_1} + \theta_{N_2}\mathcal{P}^{(IDLE)}_{NET-N_2}. \tag{25}$$

In the above formulas, we have that: (i) $\mathcal{P}^{(IDLE)}_{NET-N_1}$ (resp., $\mathcal{P}^{(IDLE)}_{NET-N_2}$) is the power (measured in (Watt)) consumed in the idle state by the NIC equipping node N_1 (resp., N_2); and, (ii) the unit step-size factor in (24) accounts for the fact that the connection: $N_1 \to N_2$ is turned ON if there is at least a task assigned to N_1 whose output data is required for the execution of at least a task assigned to N_2.

Since the involved NICs remain turned ON only during the time needed for the transport of data from node N_1 to node N_2, the dynamic component in (23) of the per-connection network energy equates, by definition, the following product:

$$\mathcal{E}^{(DYN)}_{N_1\to N_2} = \mathcal{P}^{(DYN)}_{N_1\to N_2} \times T_{N_1\to N_2}. \tag{26}$$

In the above relationship, we have that: (i) $\mathcal{P}^{(DYN)}_{N_1\to N_2}$(Watt) is the dynamic part of the network power needed for sustaining the connection: $N_1 \to N_2$; and, (ii) $T_{N_1\to N_2}$ (s) is the total time needed for the transport of the required data from N_1 to N_2. By design, it is given by:

$$T_{N_1\to N_2} \stackrel{\text{def}}{=} \frac{vl_{N_1\to N_2}}{R_{N_1\to N_2}} \equiv \frac{\left(1+\overline{NF}_{N_1\to N_2}\right)}{R_{N_1\to N_2}} \times \sum_{i=1}^{V}\sum_{j=1}^{V} a_{ij}d_{ij}\delta\left(x_i - N_1\right)\delta\left(x_j - N_2\right), \tag{27}$$

where (see (15)) $vl_{N_1\to N_2}$(bit) is the total volume of data to be transported from node N_1 to node N_2.

After swapping the indexes N_1 and N_2, the same formulas also hold for modeling the energy $\mathcal{E}_{N_2\to N_1}$ consumed by the one-way connection $N_2 \to N_1$ in (22).

1) MODELING THE DYNAMIC ENERGY WASTED BY THROUGHPUT-ADAPTIVE UP-DOWN WIRELESS CONNECTIONS

The above formulas for the network energy apply to the evaluation of the set of energy: $\{\mathcal{E}_{M\leftrightarrow N}, N \in BHS\}$ of the two-way up/down single-hop connections: $\{M \leftrightarrow N, N \in BHS\}$ embraced by the FRAN and CRAN of Fig. 1. For this purpose, it suffices that the corresponding dynamic components $\left\{\mathcal{P}^{(DYN)}_{M\to N}\right\}$ of the one-way network power in (26) are suitably modeled, in order to account for the dependence of the transmit and receive dynamic power $\mathcal{P}^{(DYN;Tx)}_{M\to N}$ and: $\mathcal{P}^{(DYN;Rx)}_{M\to N}$ on the underlying transport throughput: $R_{(M\to N)}$ (bit/s). In this regard, the results reported, for example, in [40], support the adoption of the following quite general power-like model:

$$\mathcal{P}^{(DYN)}_{M\to N} = \underbrace{\theta_M\Omega^{(Tx)}_{(M,N)} \times \left(R_{M\to N}\right)^{\xi^{(Tx)}_{(M,N)}}}_{\mathcal{P}^{(DYN;Tx)}_{M\to N}} + \underbrace{\theta_N\Omega^{(Rx)}_{(M,N)} \times \left(R_{M\to N}\right)^{\xi^{(Rx)}_{(M,N)}}}_{\mathcal{P}^{(DYN;Rx)}_{M\to N}}. \tag{28}$$

In the above equation, the (non-negative dimensionless) transmit/receive exponents: $\xi^{(Tx)}_{(M,N)} \geq \xi^{(Rx)}_{(M,N)} \geq 2$ depend on the (short or long-range) single-hop wireless communication technology adopted to sustain the connection $M \to N$. Furthermore, the (positive) transmit/receive coefficients: $\Omega^{(Tx)}_{(M,N)}$ and $\Omega^{(Rx)}_{(M,N)}$ (measured in $\left(\text{Watt}/\left((\text{bit/s})^{\xi} \times (\text{s})^{\eta}\right)\right)$) account for the effects of the Round-Trip-Time (RTT), spatial range and power profile of the considered connection. According, for example, to [3] and references therein, they may be modeled as follows:

$$\Omega^{(Tx)}_{(M,N)} = \frac{\left(RTT_{(M,N)}\right)^{\eta}\chi^{(Tx)}_{(M,N)}}{\left(1+\left(\ell_{(M,N)}\right)^{\alpha}\right)}, \text{ and}$$
$$\Omega^{(Rx)}_{(M,N)} = \frac{\left(RTT_{(M,N)}\right)^{\eta}\chi^{(Rx)}_{(M,N)}}{\left(1+\left(\ell_{(M,N)}\right)^{\alpha}\right)}, \tag{29}$$

where: (i) $RTT_{(M,N)}$ (s) is the round-trip-time of the sustained TCP/IP connection; (ii) η is a dimensionless non-negative shaping exponent (i.e., typically, $\eta \cong 0.6$); (iii) $\ell_{(M,N)}$ (m) is the physical length (i.e., the spatial range) spanned by the considered connection; (iv) α (with $2 < \alpha \leq 4$) is the fading-induced loss exponent; and, (v) the positive coefficients: $\chi^{(Tx)}_{(M,N)}$ and $\chi^{(Rx)}_{(M,N)}$ account for the transmit/receive power efficiency of the adopted wireless communication technology. As a consequence, their actual values depend on a number of communication parameters [3], [41], like, number of transmit/receive antennas [42], [43], antenna gains, coding gains, implemented carrier tracker and interleaving depth [44], just to cite a few.

2) ENERGY MODELS FOR THE TWO-WAY BACKHAUL CONNECTIONS

The general formulas reported in (22)-(27) apply verbatim to model the energy $\mathcal{E}_{N_1\leftrightarrow N_2}$ consumed by the end-to-end two-way Transport-layer connection between any pair of backhaul nodes $N_1, N_2 \in BHS$. Hence, in order to complete the corresponding model, it suffices to detail the expressions assumed by the involved dynamic network power $\mathcal{P}^{(DYN)}_{N_1\to N_2}$ in (26) and the related backhaul transport throughput $R_{N_1\to N_2}$.

In this regard, three main remarks are in order. First, each backhaul connection is symmetric, full-duplex and it may be multi-hop. Second, since it typically uses a broadband

Ethernet-type technology at the Data Link Layer, its steady-state transport capacity may be considered nearly constant over long time intervals, so that the dependence of $\mathcal{P}_{N_1\to N_2}^{(DYN)}$ on $R_{N_1\to N_2}$ is not a critical issue [36]. As a consequence, $\mathcal{P}_{N_1\to N_2}^{(DYN)}$ may be considered fixed and given by the following product formula:

$$\mathcal{P}_{N_1\to N_2}^{(DYN)} \equiv \mathcal{P}_{N_2\to N_1}^{(DYN)} = no_{HOP}^{(N_1,N_2)} \times \mathcal{P}_{HOP}^{(N_1\to N_2)} \times \max\{\theta_C, \theta_F\}, \quad N_1 \neq N_2;\ N_1, N_2 \in BHS, \tag{30}$$

where: (i) $no_{HOP}^{(N_1,N_2)}$ is the (integer-valued and positive) number of hops of the considered backhaul connection; (ii) $\mathcal{P}_{HOP}^{(N_1\to N_2)}$ (Watt) is the one-way per-hop consumed power; and, (iii) the max factor in (30) accounts for the fact that the Cloud and Fog Providers may apply different pricing policies, so that, in general, the mobile user is charged by the resources wasted by the utilized backhaul connection if at least one Provider prices them [38]. Third, in general, a TCP/IP-based backhaul connection is not so affected by the mobility of the served Mobile device and, then, it tends to operate in the Congestion Avoidance state (i.e., in the steady-state) over large time intervals [36]. As a consequence, its transport throughput may be considered nearly constant and given by the following (quite usual) formula (see, for example, [36, Chapter 7]):

$$R_{N_1\to N_2} \equiv R_{N_2\to N_1} = \frac{1.22 \times MSS^{(N_1,N_2)}}{RTT_{(N_1,N_2)} \times \sqrt{Pr_{LOSS}^{(N_1,N_2)}}}, \tag{31}$$

where: (i) $MSS^{(N_1,N_2)}$ (bit) is the maximum segment size of the considered TCP/IP connection; and, (ii) $RTT_{(N_1,N_2)}$ (s) and $Pr_{LOSS}^{(N_1,N_2)}$ are the corresponding round-trip-time and average segment loss probability.

VI. THE CONSIDERED JOINT OPTIMIZATION PROBLEM (JOP)

The goal of this section is threefold. First, by referring to the peculiar features of the mobile stream applications, we discuss the objective design targets to be pursued and the main constraints to be considered. Second, on the basis of them, we formally introduce the optimization problem to be tackled with, and, then, we consider its feasibility. Third, after pointing out the challenges presented by its solution, we develop an (optimality-preserving) decomposition of the stated optimization problem into the cascade of two inter-related sub-problems that are simpler to manage.

A. DESIGN TARGETS, RELATED CONSTRAINTS AND AFFORDED PROBLEM

By design, mobile stream applications refer to execution environments where [1]: (i) the applications are described by DAGs that are submitted for the execution in an online way (i.e., their submissions are sequential over the time); (ii) forecasting about future submissions is not available; (iii) the most relevant performance metric is the application throughput: $TH_{DAG} \stackrel{\text{def}}{=} (1/T_{DAG})$ (app/sec), that measures the rate at which the sequence of input DAGs is processed by the networked computing infrastructure of Fig. 1; and, (iv) the energy consumption of the overall ecosystem of Fig. 1 must be as low as possible.

On the basis of this native features, we identify four main design targets to be met.

First, we must guarantee that the application throughput TH_{DAG} of the underlying stream application does not fall *below* an assigned minimum value [1]: $TH_0^{(MIN)} \stackrel{\text{def}}{=} \left(1/T_{DAG}^{(MAX)}\right)$ (app/sec), that, in turn, is dictated by the QoS level to be provided to the mobile user.

Second, since Cloud and Fog providers aim to maximize the number of the users who are concurrently served and the container-based virtualization technology allows isolation of the computing resources on a per-user basis [8], the per-core computing frequency available at each computing node N must be considered limited up to a maximum value: $f_N^{(MAX)}$ (bit/sec), that, in turn, is dictated by service policy of the Service Providers.

Third, since 5G technology allows virtualization and isolation of the wireless network resources done available to each user by the FRAN and CRAN of Fig. 1 (see Table 1), both the per-connection up and down network throughput must be considered upper limited up to: $R_{M\to N}^{(MAX)}$ (bit/sec), and $R_{N\to M}^{(MAX)}$ (bit/sec), $N \in BHS$, respectively.

Fourth, according to the so-called Green Wave pursued by both the Fog and 5G paradigms [4], the ultimate goal is the minimization of the total energy E_{TOT} (Juole) consumed by the overall ecosystem of Fig. 1 for carrying out the needed computing and network operations.

Hence, by leveraging the models detailed in Section V, this total energy is formally defined by the following relationship:

$$\mathcal{E}_{TOT} \stackrel{\text{def}}{=} \underbrace{\theta_M\mathcal{E}_M + \theta_C\mathcal{E}_C + \theta_F\mathcal{E}_{FOG}}_{\mathcal{E}_{CMP}} + \underbrace{\mathcal{E}_{SR-WNET} + \mathcal{E}_{LR-WNET} + \mathcal{E}_{BH-NET}}_{\mathcal{E}_{NET}}. \tag{32}$$

In (32), we have that (see Section V):

i. $\mathcal{E}_{CMP}$ (resp., $\mathcal{E}_{NET}$) is the total energy consumed by the overall ecosystem of Fig. 1 for sustaining the computing (resp., networking) operations needed for a single execution of the considered application DAG;
ii. $\mathcal{E}_M$ and $\mathcal{E}_C$ are the computing energy wasted by the mobile device and cloud clone, respectively (see Section V-B for their formal models);
iii. $\mathcal{E}_{FOG}$ (Juole) is the summation of the computing energy wasted by the device clones deployed at the Fog nodes. Hence, it is formally defined as follows:

$$\mathcal{E}_{FOG} \stackrel{\text{def}}{=} \sum_{l=1}^{Q} \mathcal{E}_{F_l}, \tag{33}$$

where $\mathcal{E}_{F_l}$ (Juole), $l = 1, \ldots, Q$, is the computing energy wasted by the device clone at the l-th Fog node (see Section V-B for its formal model);

iv. $\mathcal{E}_{SR-WNET}$ (Juole) is the total network energy wasted by the two-way (i.e., up/down) Short Range (SR) single-hop wireless connections embraced by the FRAN of Fig. 1. It is formally defined as:

$$\mathcal{E}_{SR-WNET} \stackrel{\text{def}}{=} \sum_{l=1}^{Q} \mathcal{E}_{M\leftrightarrow F_l}, \tag{34}$$

where: $\mathcal{E}_{M\leftrightarrow F_l}$ (Juole), $l = 1, \ldots, Q$, is the network energy consumed by the wireless up/down single-hop connection between the Mobile device and the l-th Fog node (see Section V-C for its formal model);

v. $\mathcal{E}_{LR-WNET} \stackrel{\text{def}}{=} \mathcal{E}_{M\leftrightarrow C}$ (Juole) is the energy consumed by the CRAN of Fig. 1, in order to sustain the two-way (i.e., up/down) Long-Range (LR) single-hop cellular connection between the Mobile device and the remote Cloud (see Section V-C for its formal model); and, finally,

vi. $\mathcal{E}_{BH-NET}$ (Juole) is the total energy wasted by all the Fog-to-Fog (F2F) and Fog-to-Cloud (F2C) two-way (possibly, multi-hop) Transport-layer connection embraced by the Backhaul network segment of Fig. 1. It reads as:

$$\mathcal{E}_{BH-NET} \stackrel{\text{def}}{=} \underbrace{\sum_{l=1}^{Q} \sum_{\substack{k=1 \\ k>l}}^{Q} \mathcal{E}_{F_l\leftrightarrow F_k}}_{\text{F2F Network Energy}} + \underbrace{\sum_{l=1}^{Q} \mathcal{E}_{F_l\leftrightarrow C}}_{\text{F2C Network Energy}} \tag{35}$$

where the energy present in the above summations are modeled as reported in the last part of Section V-C.

Let:

$$\overrightarrow{RS} \stackrel{\text{def}}{=} \left[f_N, N \in A; R_{M\rightarrow N}, R_{N\rightarrow M}, N \in BHS\right] \in (\mathbb{R}_+)^{3Q+4}, \tag{36}$$

be the $(3Q+4)$-dimensional (row) vector that collects the processing frequencies of the device clones at the computing nodes and the up-plus-down wireless transport throughput over the FRAN and CRAN of Fig. 1. Therefore, from the outset, it follows that the afforded *Joint Optimization Problem* (JOP) may be formally defined as the following mixed-integer constrained optimization problem:

$$\min_{\vec{x},\overrightarrow{RS}} \mathcal{E}_{TOT}\left(\vec{x}, \overrightarrow{RS}\right), \tag{37a}$$

$$\text{s.t.: } T_{DAG} \leq \frac{1}{TH_0^{(MIN)}} \equiv T_{DAG}^{(MAX)}, \tag{37b}$$

$$0 \leq f_N \leq f_N^{(MAX)}, \quad N \in \mathcal{A}, \tag{37c}$$

$$0 \leq R_{M\rightarrow N} \leq R_{M\rightarrow N}^{(MAX)}, \quad N \in BHS, \tag{37d}$$

$$0 \leq R_{N\rightarrow M} \leq R_{N\rightarrow M}^{(MAX)}, \quad N \in BHS, \tag{37e}$$

$$x_1 = x_V = M, \tag{37f}$$

$$x_i \in \left\{M, F_1, \ldots, F_Q, C\right\}, \quad i = 2, \ldots, (V-1). \tag{37g}$$

In the reported *JOP* formulation, we have that: (i) the objective function in (37a) is given by the (weighted) summation of the computing and network energy in (32). Hence, as stressed in (37a), its actual value jointly depends on the task and resource allocation vectors $\vec{x}$ and $\overrightarrow{RS}$, that, in turn, play the role of optimization variables; (ii) the constraint in (37b) guarantees that the per-DAG execution time meets the (previously mentioned) bound on the minimum throughput required by the processed stream application; (iii) the box constraints in (37c), (37d) and (37e) account for the (aforementioned) limitations on the allowed computing and network resources; (iv) the constraints in (37f) account for the fact that the first and last tasks of the DAG must be executed by the Mobile device (see the assumptions of Section III on the application DAG); and, (v) the last group of constraints in (37g) forces each scalar component of the task allocation vector to take only and only one value over the discrete set: $\left\{M, F_1, \ldots, F_Q, C\right\}$.

Before proceeding, two main remarks on the practical impacts of the reported *JOP* formulation and the flexibility of the developed formal framework are in order.

Remark 1 (On the Flexibility of the Developed Formal Framework): About the flexibility of the developed system framework, two main illustrative remarks may be of practical interest.

First, the developed framework featuring the FRAN, CRAN and Backhaul network segments of the overall technological platform of Fig. 1 gives rise to a *fully meshed* network topology, that, in principle, may comprise up to: $(Q+1)(Q+2)/2$ end-to-end (possibly, multi-hop) Transport-layer parallel connections. Hence, *any* overlay network topology of interest may be actually built up by deleting the connections that are not actually utilized. By referring to the reported *JOP* formulation, this may be done by: (i) setting to zero the maximum up/down throughput $R_{M\rightarrow N}^{(MAX)}$ and $R_{N\rightarrow M}^{(MAX)}$, $N \in BHS$, in the constraints of (37d) and (37e) that correspond to the wireless connections that are not really supported by the considered FRAN and/or CRAN; and, (ii) posing to zero the throughput: $R_{N_1\rightarrow N_2} \equiv R_{N_2\rightarrow N_1}$ in (31) of the two-way backhaul connections that are not instantiated by the Backhaul network segment of Fig. 1. So doing, *any* overlay network topology connecting *any number* of (possibly, hierarchically organized) Fog tiers may be built up atop the underlying networked infrastructure of Fig. 1.

Second, the Cloud and/or Fog Providers who manage the ecosystem of Fig. 1 may enforce authorization-based policies that forbid the mobile user to access the computing resources hosted by the Cloud node and/or some Fog nodes [38]. In our framework, these access limitations may be taken into account by setting to zero the maximum allowed computing frequencies $f_N^{(MAX)}$ in the *JOP* constraints of (37c) that correspond to the forbidden computing resources. ■

Remark 2 (Eco-vs.-Mobile Centric Service Models): The roles played by the (binary-valued) theta parameters previously introduced in (20) are unveiled by a direct inspection of the expressions of the network energy of Section V-C and the objective function $\mathcal{E}_{TOT}$ in (32). It points out that, by setting $\theta_N, N \in \{M, F, C\}$ at zero (resp., at the unit value), both

the computing and network energy wasted by node N *do not* contribute to the *JOP* objective function in (37a) and, then, they are *not* subject to optimization.

On the basis of these formal considerations, two SaaS models may cover practical relevance [38], namely, the Eco-centric SaaS model and the Mobile-centric one. By definition, the (emerging) Eco-centric model is defined by setting:

$$\theta_M = \theta_F = \theta_C \equiv 1, \quad (\textit{Eco-centric} \text{ SaaS model}), \quad (38)$$

while, under the (more usual) Mobile-centric model, we have that:

$$\theta_M = 1, \quad \theta_F = \theta_C \equiv 0, \quad (\textit{Mobile-centric} \text{ SaaS model}). \quad (39)$$

A comparison of the above defining relationships points out that: (i) under the Eco-centric model, the computing and network energy of *all* nodes of the ecosystem of Fig. 1 contribute to the objective function in (37a); while, (ii) under the Mobile-centric model, *only* the computing and network energy wasted by the mobile device are accounted for by *JOP* objective function.

As a consequence, it is reasonable to expect that both the task/resource allocations and the corresponding energy consumption returned by the *JOP* optimization process could be substantially different under these two service models. We anticipate the numerical results of Section X-G confirm this expectation and support the conclusion that an eco-friendly approach is capable of (drastically) reducing the total energy consumed by the overall Mobile-Fog-Cloud platform of Fig. 1 by somewhat penalizing the corresponding energy consumption experienced by the Mobile device. ■

B. JOP FEASIBILITY

The considered *JOP* is a mixed-integer non-convex optimization problem, so that its solution resists, indeed, closed-form evaluation. Due to the same reason, the derivation of closed-form analytical conditions that are both necessary and sufficient for the feasibility of the *JOP* seems to be very challenging. On the basis of these considerations, in the sequel, we focus on the derivation of a sufficient condition for the *JOP* feasibility that is in *closed-form*, and (which is the most) does *not* require the explicit pre-evaluation of the (a priori unknown) *JOP* solution.

In order to formally present this sufficient condition, some dummy positions are in order. Specifically, let:

$$s^{(MAX)} \stackrel{\text{def}}{=} \max_{1 \le i \le V} \{s_i\} \quad \text{(bit)}, \quad (40)$$

be the maximum size of the DAG tasks, and let:

$$w_{IN}^{(MAX)} \stackrel{\text{def}}{=} \max_{1 \le i \le V} \left\{ \sum_{j=1}^{V} a_{ji} d_{ji} \right\} \quad \text{(bit)}, \quad (41)$$

be the maximum volume of data that a task receives in input from the set of its preceding tasks (i.e., its parent tasks). Furthermore, let:

$$\beta^{(MIN)} = \begin{cases} 1, & \text{under the } \textit{SEQ} \text{ service discipline,} \\ \dfrac{\min\limits_{1 \le i \le V} \{\phi_i\}}{\sum_{j=1}^{V} \phi_j}, & \text{under the } \textit{WPS} \text{ service discipline,} \end{cases} \quad (42)$$

indicate the minimum fraction of the per-core computing frequency that is devoted to the execution of a single task. Hence, after denoting by:

$$Tmin_N^{(SER)} \stackrel{\text{def}}{=} \frac{s^{(MAX)}}{n_N f_N^{(MAX)} \beta^{(MIN)}}, \quad N \in \mathcal{A}, \quad (43)$$

the minimum service time at node N for the task of maximum size, let:

$$Tmax^{(SER)} \stackrel{\text{def}}{=} \max_{N \in \mathcal{A}} \left\{ Tmin_N^{(SER)} \right\} \equiv \frac{s^{(MAX)}}{\beta^{(MIN)} \times \min\limits_{N \in \mathcal{A}} \left\{ n_N f_N^{(MAX)} \right\}}, \quad (44)$$

be the maximum of the minimum service times, and let:

$$Tmax^{(NET)} \stackrel{\text{def}}{=} \frac{w_{IN}^{(MAX)} \left(1 + \max\limits_{N_1, N_2 \in \mathcal{A}} \left\{ \overline{NF}_{N_1 \to N_2} \right\} \right)}{\min\limits_{N_1, N_2 \in \mathcal{A}} \left\{ R_{N_1 \to N_2}^{(MAX)} \right\}}, \quad (45)$$

indicate the maximum of the minimum network times needed to transport the (previously defined) maximum volume of input data $w_{IN}^{(MAX)}$ used for the execution of a task. On the basis of the above positions, it follows that the maximum execution time: $T_{EXE}^{(MAX)}$ of *any* task at *any* computing node equates the summation:

$$T_{EXE}^{(MAX)} = Tmax^{(SER)} + Tmax^{(NET)}. \quad (46)$$

In the Appendix A, it is proved that a suitable exploitation of the above relationship leads to the following sufficient condition for the *JOP* feasibility.

Proposition 1 (Sufficient Condition for the JOP Feasibility): Let the previously reported assumptions on the behavior of T_{DAG} be met. Furthermore, let:

$$T_{DAG}^{(UP)} \stackrel{\text{def}}{=} T_{DAG} \left(T_i^{(EXE)} \equiv T_{EXE}^{(MAX)}, \quad i = 1, \ldots, V \right), \quad (47)$$

be the value assumed by the DAG execution time T_{DAG} under the (worst) case in which all the task execution times: $T_i^{(EXE)}$, $i = 1, \ldots, V$, equate to the maximum one $T_{EXE}^{(MAX)}$ in (46). Hence, the satisfaction of the following inequality:

$$T_{DAG}^{(UP)} \le \frac{1}{TH_0^{(MIN)}}, \quad (48)$$

suffices to guarantee the *JOP* feasibility. ■

In the sequel, we formally indicate by:

$$\left\{ \vec{x}^*, \vec{RS}^* \right\}, \quad (49)$$

the solution of a feasible *JOP*.

C. JOP DECOMPOSITION INTO RAP AND TAP

Two main formal features retained by the considered *JOP* make the analytical evaluation of its solution in (49) challenging. First, due to the presence of product terms that jointly involve a number of (scalar) components of both optimization variables $\vec{x}$ and $\overrightarrow{RS}$ (see the energy expressions in Sections V-B and V-C), the objective function $\mathcal{E}_{TOT}\left(\vec{x}, \overrightarrow{RS}\right)$ in (37a) is not jointly convex in the involved optimization variables and, then, the *JOP* is a *non-linear* and *non-convex* optimization problem. Second, the JOP embraces both real-valued $\overrightarrow{RS}$ and discrete-valued $\vec{x}$ optimization variables, so that it is a *mixed-integer* optimization problem.

Hence, in order to cope with these challenges, in the sequel, we develop an (optimality-preserving) approach, that aims to hierarchically decompose the reported *JOP* into the cascade of two inter-depending simpler optimization sub-problems, namely, the *Resource Allocation Problem (RAP)* and the *Task Allocation Problem (TAP)*. Interestingly enough, we anticipate that the *RAP* optimization involves *only* the real-valued resource allocation vector $\overrightarrow{RS}$, while the *TAP* optimization is carried out over *only* the discrete-valued variable $\vec{x}$.

Specifically, under any *assigned* task allocation vector $\vec{x}$, the *RAP* is formally defined as the following constrained optimization problem in the real-valued optimization variable $\overrightarrow{RS}$:

$$\min_{\overrightarrow{RS}} \mathcal{E}_{TOT}\left(\vec{x}; \overrightarrow{RS}\right), \tag{50a}$$

$$\text{s.t.: Eqs. (37b), (37c), (37d), (37e).} \tag{50b}$$

Let:

$$\widetilde{\overrightarrow{RS}} \equiv \widetilde{\overrightarrow{RS}}\left(\vec{x}\right), \text{ and } \widetilde{\mathcal{E}}_{TOT} \equiv \widetilde{\mathcal{E}}_{TOT}\left(\vec{x}; \widetilde{\overrightarrow{RS}}\left(\vec{x}\right)\right), \tag{51}$$

be the ($\vec{x}$-depending) solution of the *RAP* and the corresponding ($\vec{x}$-depending) value attained by the objective function in (50a) at the optimum.

Hence, the *TAP* is formally defined as the following constrained optimization problem in the discrete-valued optimization variable $\vec{x}$:

$$\min_{\vec{x}} \widetilde{\mathcal{E}}_{TOT}\left(\vec{x}; \widetilde{\overrightarrow{RS}}\left(\vec{x}\right)\right), \tag{52a}$$

$$\text{s.t.: Eqs. (37f) and (37g).} \tag{52b}$$

Let:

$$\left\{\hat{\vec{x}}, \widehat{\overrightarrow{RS}} \stackrel{\text{def}}{=} \widetilde{\overrightarrow{RS}}\left(\hat{\vec{x}}\right)\right\}, \tag{53}$$

be the solution of the *TAP* in (52a) and (52b). Then, the following Proposition 2 proves that the performed *JOP* decomposition is optimality-preserving.

Proposition 2 (On the Optimality of the Developed JOP Decomposition): Let the *JOP* be feasible. Hence, its solution in (49) coincides with the one in (53) which is obtained by cascading the solutions of the *RAP* and *TAP*, that is,

$$\vec{x}^* \equiv \hat{\vec{x}}, \text{ and } \overrightarrow{RS}^* \equiv \widehat{\overrightarrow{RS}}. \tag{54}$$

Proof: The proof of (54) relies on the following three formal properties retained by the performed *JOP* decomposition.

First, without loss of optimality, the joint minimum in (37a) may be decomposed into the cascade of the following two minima [37]:

$$\min_{\vec{x}, \overrightarrow{RS}} \mathcal{E}_{TOT}\left(\vec{x}, \overrightarrow{RS}\right) \equiv \min_{\vec{x}} \left\{\min_{\overrightarrow{RS}} \mathcal{E}_{TOT}\left(\vec{x}, \overrightarrow{RS}\right)\right\}. \tag{55}$$

Second, all the constraints of the *JOP* formulation in (37b), (37c), (37d), and (37e) (resp., in (37f) and (37g)) that involve the resource allocation vector $\overrightarrow{RS}$ (resp., the task allocation vector $\vec{x}$) are taken into account in the *RAP* formulation (resp., the *TAP* formulation).

Third, for any assigned task allocation vector $\vec{x}$, the *TAP*'s objective function: $\widetilde{\mathcal{E}}_{TOT}\left(\vec{x}; \widetilde{\overrightarrow{RS}}\left(\vec{x}\right)\right)$ in (52a) explicitly accounts for the optimal solution $\widetilde{\overrightarrow{RS}}\left(\vec{x}\right)$ returned by the *RAP*. □

VII. THE DEVELOPED RAP SOLVING APPROACH

The goal of this section is threefold. First, we develop the formal conditions under which the *RAP* is convex and feasible. Second, we develop a formal approach for computing the *RAP* solution in (50) that is adaptive and capable to self-react to the mobility-induced changes of the operating environment of the ecosystem of Fig. 1. Third, we point out some aspects related to the implementation and implementation complexity of the developed *RAP* solving approach.

Specifically, in Appendix C, the following conditions for the convexity of the *RAP* are proved.

Proposition 3 (On the Convexity of the RAP): Let the assumptions of Section IV-C on T_{DAG} be met. Furthermore, let us assume that the exponents of the dynamic computing and network power in (21) and (28) meet the following inequalities:

$$\gamma_N \geq 2, \quad N \in \mathcal{A}, \tag{56}$$

and

$$\xi_{(M,N)}^{(Tx)} \geq 2, \ \xi_{(M,N)}^{(Rx)} \geq 2, \quad N \in BHS. \tag{57}$$

Then, the *RAP* in (50a), (50b) is *convex* in the resource allocation vector $\overrightarrow{RS}$ under any assigned task allocation vector $\vec{x}$. ■

Passing to consider the *RAP* feasibility, let:

$$\overrightarrow{RS}^{(MAX)} \stackrel{\text{def}}{=} \left[f_N^{(MAX)}, N \in A; R_{M\to N}^{(MAX)}, R_{N\to M}^{(MAX)}, N \in BHS\right] \in \left(\mathbb{R}_+\right)^{3Q+4}, \tag{58}$$

indicate the vector of the maximal allowed resources. Hence, in Appendix D, the following necessary and sufficient condition is proved.

Proposition 4 (On the Feasibility of the RAP): Let the assumptions of Section IV-C on T_{DAG} be met. Then, the *RAP*

in (50a), (50b) is feasible *if and only if* the following inequality holds:

$$TH_0^{(MIN)} \times T_{DAG}\left(\vec{x}; \overrightarrow{RS}^{(MAX)}\right) - 1 \leq 0, \tag{59}$$

where $T_{DAG}\left(\vec{x}; \overrightarrow{RS}^{(MAX)}\right)$ indicates the value assumed by the DAG execution time under the task and resource allocation vectors $\vec{x}$ and $\overrightarrow{RS}^{(MAX)}$, respectively. ■

Before proceeding, two explicative remarks are in order concerning the conditions reported in the above two Propositions 3 and 4. First, thereinafter we assume that the inequalities in (56), (57) are met, so that the considered *RAP* is guaranteed to be convex. Second, since the satisfaction of the inequality in (59) may depend on the considered $\vec{x}$, in the sequel, we label a task allocation vector $\vec{x}$ as *RAP-feasible* if it meets the *RAP* feasibility condition in (59).

A. FEATURING THE RAP SOLUTION

Let us assume that the considered *RAP* is convex and feasible. Hence, the general results reported, for example, by *Theorems 4.3.7* and *4.3.8* of [37] guarantee that the Karush-Kuhn-Tucker (KKT) conditions are both necessary and sufficient for the evaluation of the *RAP* solution, provided that the *RAP* also meets the so-called Slater's qualification condition (see, for instance, [37, Chapter 5]). In this regard, in Appendix E, the following formal result is proved.

Proposition 5 (Slater's Qualification for the RAP): Let us assume that the feasibility condition in (59) is met with the *strict* inequality under the considered task allocation vector $\vec{x}$. Then, the Slater's qualification holds for the *RAP* in (50a), and (50b). ■

Under the assumption that the condition of Proposition 5 is met, we resort to a suitable application of the so-called *Primal-Dual Solving Approach (PDSA)*, in order to evaluate the *RAP* solution (see, for example, Chapter 10 of [37]). In this regard, we note that the *RAP* convexity allows to manage the box constraints in (37c)-(37e) on the maximal resources as implicit ones. Hence, under any assigned task allocation vector $\vec{x}$, the resulting Lagrangian function $\mathcal{L}\left(\overrightarrow{RS}, \lambda\right)$ (Joule) of the *RAP* reads as follows:

$$\mathcal{L}\left(\overrightarrow{RS}, \lambda\right) \stackrel{\text{def}}{=} \mathcal{E}_{TOT}\left(\vec{x}; \overrightarrow{RS}\right) + \lambda\left(TH_0^{(MIN)} \times T_{DAG}\left(\vec{x}; \overrightarrow{RS}\right) - 1\right), \tag{60}$$

where the (scalar and non-negative) parameter λ (Joule) is the Lagrange multiplier associated to the (convex) constraint in (37b). Therefore, from a formal point of view, the constrained max-min optimization problem to be solved for the evaluation of the *RAP* vector solution $\widetilde{\overrightarrow{RS}}$ in (51) and the corresponding optimal Lagrange multiplier $\tilde{\lambda}$ is the following one (see, for example, [37, Chapter 6]):

$$\max_{\lambda \geq 0}\left\{\min_{\vec{0} \leq \overrightarrow{RS} \leq \overrightarrow{RS}^{(MAX)}}\left\{\mathcal{L}\left(\overrightarrow{RS}, \lambda\right)\right\}\right\}, \tag{61}$$

where $\overrightarrow{RS}^{(MAX)}$ is the vector of the maximum available resources in (58).

Now, let $\overrightarrow{\nabla}\mathcal{L}\left(\overrightarrow{RS}, \lambda\right)$ be the $(3Q+5)$-dimensional gradient vector of the Lagrangian function in (60) performed with respect to the $(3Q+4)$ scalar components of the resource vector $\overrightarrow{RS}$ (i.e., the vector of the so-called primal variables) and the (scalar) Lagrange multiplier λ (i.e., the so-called dual variable). Hence, *Theorem 6.2.6* of [37] guarantees that the solution of the max-min problem in (61) (that is, the so-called saddle-point of the considered Lagrangian function) may be computed by performing the orthogonal projection onto the box set: $\left[\vec{0}, \overrightarrow{RS}^{(MAX)}\right]$ of the allowed resources of the vector solution of the following $(3Q+5)$-dimensional algebraic equation system:

$$\overrightarrow{\nabla}\mathcal{L}\left(\overrightarrow{RS}, \lambda\right) = \vec{0}. \tag{62}$$

B. DEVELOPED ADAPTIVE SOLVING APPROACH

Two main remarks about the solution:

$$\left\{\widetilde{\overrightarrow{RS}}, \tilde{\lambda}\right\}, \tag{63}$$

to the system of equations in (62) are in order. First, since the *RAP* objective function $\mathcal{E}_{TOT}$ in (50a) and the constraint in (37b) on the allowed T_{DAG} are nonlinear functions of the optimization variable $\overrightarrow{RS}$, the resulting system of algebraic equations in (62) is nonlinear, and, in general, its solution resists closed-form evaluation. Second, even if it would possible to solve in closed-form the algebraic equations in (62) under some specific cases, the obtained closed-form solution should be re-evaluated *from scratch* when the device mobility and/or the occurrence of network congestion phenomena induce abrupt (and, typically unpredictable) changes in the operating environment of the ecosystem of Fig. 1.

Therefore, in the sequel, we develop an iteration-based solving approach, in order to allow the solution in (63) of (62) to *self-react* to environmental changes. Specifically, the pursued approach allows us to iteratively evaluate the solution of (62) through the adaptive implementation of a suitable set of projected gradient-based primal-dual scaled iterations. As pointed out in [37], the primal-dual algorithm is an iterative procedure that updates on a per-step basis both the primal variables (i.e., the optimization variables) and the dual ones (i.e., the Lagrange multipliers associated to the underlying constraints), in order to guide the corresponding Lagrangian function towards its saddle-point. Hence, after introducing the dummy position:

$$[z]_a^b \stackrel{\text{def}}{=} \max\{a, \min\{z, b\}\}, \tag{64}$$

the $(m+1)$-th updating of the l-th scalar component: y_l, $l = 1, \ldots, (3Q+4)$, of the resource vector $\overrightarrow{RS}$ reads as in:

$$y_l^{(m+1)} = \left[y_l^{(m)} - \psi_l^{(m)}\left(\frac{\partial\mathcal{L}\left(\overrightarrow{RS}^{(m)}, \lambda^{(m)}\right)}{\partial y_l}\right)\right]_0^{y_l^{(MAX)}}, \tag{65}$$

and the $(m+1)$-th updating of the Lagrange multiplier is dictated by the following iteration:

$$\lambda^{(m+1)} = \left[\lambda^{(m)} + \xi^{(m)} \frac{\partial \mathcal{L}\left(\overrightarrow{RS}^{(m)}, \lambda^{(m)}\right)}{\partial \lambda} \right]_0^{+\infty}. \quad (66)$$

In the above iterations, we have that: (i) $m \geq 0$ is an integer-valued iteration index; (ii) $\partial \mathcal{L}(\overrightarrow{RS}^{(m)}, \lambda^{(m)})/\partial y_l$ (resp., $\partial \mathcal{L}(\overrightarrow{RS}^{(m)}, \lambda^{(m)})/\partial \lambda$) is the partial derivative of the Lagrangian function in (60) performed with respect to the l-th (scalar) component y_l of the resource vector $\overrightarrow{RS}$ (resp., with respect to the λ multiplier) and evaluated at iteration m; (iii) $y_l^{(MAX)}$ in (65) is the corresponding maximum value allowed y_l (that is, the l-th scalar component of the maximal resource vector $\overrightarrow{RS}^{(MAX)}$ in (58)); and, (iv) $\left\{\psi_l^{(m)}, l = 1, \ldots, (3Q+4)\right\}$ and $\xi^{(m)}$ are non-negative time-variyng (i.e., m-varying) step-sizes. Furthermore, according to the max-min saddle-point relationship in (61), the minus (resp., plus) sign is present in (65) (resp., (66)), so that (65) (resp., (66)) features descending-gradient (resp., ascending-gradient) iterations.

C. DESIGN OF THE TIME-VARYING STEP-SIZES AND CONVERGENCE PROPERTY

The peculiar feature shared by the iterations in (65) and (66) is that they resort to time-varying step-sizes, in order to guarantee *fast adaptation* of the corresponding primal and dual variables in response to abrupt changes of the operating environment of the ecosystem of Fig. 1. For this purpose, the approach quite recently reported in [6] may be pursued. Specifically, two main formal results of [6] are relevant in our context. First, *Theorem 3.3* of [6] proves that it is sufficient to update each step-size sequence on the basis of *only* the corresponding primal-dual variable, in order to guarantee the asymptotic convergence to the global optimum of *all* iterations in (65) and (66). Second, a suitable choice for updating each step-size is to set it proportionally to the *squared value* of the corresponding primal-dual variable at each iteration step m.

Hence, motivated by these formal results, we planned to implement the following "up/down clipped" relationships for the updating of the step-sizes in (65) and (66):

$$\psi_l^{(m)} = \max\left\{a_{MAX}, \min\left\{a_{MAX} \times y_l^{(MAX)}, \left(y_l^{(m)}\right)^2\right\}\right\}, \quad (67)$$

and

$$\xi^{(m)} = \max\left\{a_{MAX}, \min\left\{a_{MAX} \times \max_l\left\{y_l^{(MAX)}\right\}, \left(\lambda^{(m)}\right)^2\right\}\right\}. \quad (68)$$

Interestingly enough, the role played in (67) and (68) by the introduced clipping factor: a_{MAX} is twofold. First, (i) it allows a fast reaction in response to abrupt (possibly, unpredicted and mobility induced) environmental changes; and, (ii) it speeds up the convergence to the global optimum of the iterations in (65), (66) by forbidding too small step-size values (see the outer max(.) in (67), (68)). Second, it avoids too strong oscillations of the underlying iterations around their steady-state values by clipping the maximum value allowed by each step-size (see the inner min(.) in (67), (68)). We anticipate that the numerical results of Section X-B support the actual effectiveness of the performed design and also give practical insights about the right setting of the clipping factor a_{MAX}.

D. IMPLEMENTATION ASPECTS AND IMPLEMENTATION COMPLEXITY OF THE RAP ITERATIONS

The pseudo-code of Algorithm 1 details the ordered list of steps that are needed for the software implementation of the *RAP* iterations in (65) and (66). In a nutshell, the pseudo-code receives in input the task allocation vector $\vec{x}$ to be processed, and, after checking the *RAP* feasibility condition of Proposition 4, it performs I_{MAX} runs of the primal-dual iterations in (65), (66). After completing the I_{MAX}-th run, it returns both the attained resource allocation vector $\widetilde{\overrightarrow{RS}}$ and the associated consumed energy $\widetilde{\mathcal{E}}_{TOT}$ of (51). If the *RAP* would be infeasible, the code of Algorithm 1 sets all the returned outputs at the infinite and halts its execution (see step 2 of Algorithm 1).

Therefore, a direct inspection of this pseudo-code leads to three main insights about the related implementation complexity. First, the implementation complexity of the developed *RAP* solving approach is *fully independent* of the size of the underlying application DAG. Second, since, at each run, it scales with the number of updated primal-dual variables as (see (65) and (66)) $\mathcal{O}(3Q+5)$, the overall implementation complexity of the *RAP* solving approach of Algorithm 1 over I_{MAX} runs scales as:

$$\mathcal{O}(I_{MAX} \times (3Q+5)). \quad (69)$$

Third, the above relationship points out that the scaling behavior is *linear* with respect to both the number I_{MAX} of carried out runs, and the number $(Q+2)$ of computing nodes of the ecosystem of Fig. 1.

VIII. THE DEVELOPED TAP SOLVING APPROACH

The solution in (53) of the *TAP* in (52a) and (52b) resists closed-form evaluation due to the following three main reasons. First, the *TAP* is a discrete optimization problem in the task allocation vector $\vec{x}$ in (6), so that, unlike the *RAP*, its solution cannot be approached by resorting to the (aforementioned) gradient-based KKT conditions. Second, due to the presence of product terms of unit-step functions that involve multiple scalar components of the optimization variable $\vec{x}$, the resulting *TAP* objective function: $\widetilde{\mathcal{E}}\left(\vec{x}; \widetilde{\overrightarrow{RS}}(\vec{x})\right)$ in (52a) is not convex. Hence, numerical routines for the evaluation of the solution of convex discrete optimization problems (as, for example, CVX, i.e., the MATLAB software for disciplined Convex Programming) cannot be applied.

Algorithm 1 Computing the *RAP* Solution

Input: Task allocation vector $\vec{x}$.

Output: Resource allocation vector $\widetilde{\overrightarrow{RS}}$;
Associated consumed energy $\widetilde{\mathcal{E}}_{TOT}$.

▷ Feasibility test

1: **if** the *RAP* feasibility condition in (59) fails **then**
2: Set $\widetilde{\overrightarrow{RS}}$ and $\widetilde{\mathcal{E}}_{TOT}$ to infinite;
3: **return** $\widetilde{\overrightarrow{RS}}$ and $\widetilde{\mathcal{E}}_{TOT}$;
4: **end if**

▷ Iterative phase

5: **for** $m = 0 : (I_{MAX} - 1)$ **do**
6: Compute the set of Lagrangian derivatives involved by (65) and (66);
7: Compute the step-sizes in (67) and (68);
8: Update the set of resource variables $\left\{ y_l \in \widetilde{\overrightarrow{RS}} \right\}$ through (65);
9: Update the Lagrange multiplier λ through (66);
10: Update $\widetilde{\mathcal{E}}_{TOT}$ through (32);
11: **end for**
12: **return** $\widetilde{\overrightarrow{RS}}$ and $\widetilde{\mathcal{E}}_{TOT}$.

On the basis of these considerations, in the remaining part of this section, we address these challenges by developing a version of the Genetic Algorithm (GA) that is devised on an *ad hoc* basis, tailored to the peculiar features of the *TAP* to be solved.

A. THE PROPOSED ADAPTIVE GENETIC-BASED SOLVING APPROACH

In the last years, the GA paradigm has been applied with good success for approaching the solutions of a number of task scheduling, resource consolidation and resource migration discrete optimization problems under various delay and energy-induced constraints (see, for example, [45] and [46, Chapter 8]).

Roughly speaking, the GA is a meta-heuristic for iterative search, that simulates the natural evolution process. In general, the GA requires that a population (of size *PS*) of discrete-valued vectors $\{\vec{x}_k, k = 1, \ldots, PS\}$, representing candidate solutions, evolves towards better solutions by leveraging suitable crossover and mutation functions. The goodness of each candidate: $\vec{x}_k$ is measured through an (application-depending) fitness function $fit\ (\vec{x}_k)$. At each generation, the fitness of each element of the current population is evaluated and a set of elements is selected by comparing their fitness. Specifically, a fraction *CF* of the "*best*" elements of the current population is recombined through crossover, while the remaining "worst" elements are modified by randomly mutating each one over *MN* randomly selected positions. Therefore, after selecting the "best" *PS* elements from the obtained set of all crossed over and mutated elements, a new population is formed. This last is used in the next iteration of the GA. The GA algorithm halts when either a maximum number G_{MAX} of generations (i.e., iterations) is carried out, or a good fitness level is reached by the *best* element $\vec{x}_{BEST}$ of the generated populations.

By design, the *Adaptive Genetic Task Allocation Strategy (A-GTA-S)* proposed in this paper for approaching the *TAP* solution retains the following three main characteristics.

First, each individual $\vec{x}$ of a population is a V-tuple task allocation vector, that, in turn, is defined according to (6).

Second, the fitness $fit\ (\vec{x})$ of each individual equates to the inverse of the energy $\widetilde{\mathcal{E}}_{TOT}\ (\vec{x}) \equiv \widetilde{\mathcal{E}}_{TOT}(\vec{x}; \widetilde{\overrightarrow{RS}}(\vec{x}))$ in (51) returned by the solution of the *RAP*, that is,

$$fit\ (\vec{x}) \stackrel{\text{def}}{=} 1/\widetilde{\mathcal{E}}_{TOT}\ (\vec{x}), \quad \left(\text{Joule}^{-1}\right). \tag{70}$$

This definition reflects the fact that, in our framework, the "best" task allocations are those that require less energy, in order to be sustained.

Third, from a formal point of view, the proposed *A-GTA-S* is an example of *elitary* GA [46], i.e., it guarantees that the final returned solution $\vec{x}_{BEST}$ is the *global best* over the set of all computed G_{MAX} generations.

Pseudo-Code of the Proposed A-GTA-S: A pseudo-code of the proposed *A-GTA-S* is reported in Algorithm 2. In order to facilitate its actual software implementation, the reported pseudo-code details also the main required data structures, i.e., the (dummy) matrices [*Poplist*], [*Childlist*] and [*Mutationlist*] of Algorithm 2. According to the GA description reported at the beginning of this section, the input data of Algorithm 2 are the (previously introduced) parameters: *PS*, *CF*, G_{MAX} and *MN*, while the returned output is the triplet:

$$\left[\vec{x}_{BEST},\ \overrightarrow{RS}_{BEST},\ \mathcal{E}_{BEST}\right], \tag{71}$$

of the task allocation vector, resource allocation vector and associated consumed energy. In the sequel, we refer to the triplet in (71) as the solution of the *A-GTA-S*.

Algorithm 3 and Algorithm 4 detail the related pseudo-codes of the *Crossover* and *Mutation* functions called by *A-GTA-S* at steps 9 and 11 of Algorithm 2.

Algorithm 2 Pseudo-Code of the Proposed A-GTA-S

Input: Population size PS;
Fraction CF of the crossed over population;
Number G_{MAX} of performed generations;
Number MN of mutated components.

Output: Best task allocation vector $\vec{x}_{BEST}$;
Best resource allocation vector $\overrightarrow{RS}_{BEST}$;
Best consumed energy $\mathcal{E}_{BEST}$.

▷ Initialization phase

1: Generate a random initial list: $\{\vec{x}_k, k = 1, \ldots, PS\}$ of task allocation vectors and store it into the (dummy) matrix $[Poplist]$ on a per-row basis;
2: **for** $k = 1 : PS$ **do**
3: Compute $\widetilde{\overrightarrow{RS}}(\vec{x}_k)$ and: $\widetilde{\mathcal{E}}_{TOT}\left(\vec{x}_k; \widetilde{\overrightarrow{RS}}(\vec{x}_k)\right)$ in (51) by running Algorithm 1 under $\vec{x}_k \in [Poplist]$;
4: Store the obtained $\widetilde{\overrightarrow{RS}}(\vec{x}_k)$ and $\widetilde{\mathcal{E}}_{TOT}\left(\vec{x}_k; \widetilde{\overrightarrow{RS}}(\vec{x}_k)\right)$ into the k-th row of the matrix $[Poplist]$;
5: **end for**
6: Sort the row of the matrix $[Poplist]$ for increasing values of the energy of its elements;
7: Set the number $Cross \stackrel{\text{def}}{=} [CF \times PS]$ of the elements of the matrix $[Poplist]$ to be crossed over at each generation;

▷ Iterative phase

8: **for** $j = 1 : G_{MAX}$ **do**
9: Perform the pair-wise crossover of the first $Cross$ elements of the matrix $[Poplist]$ by calling $(Cross/2)$ times the *Crossover* function in Algorithm 3;
10: Store the obtained crossed over elements in the (dummy) matrix $[Childlist]$ on a per-row basis;
11: Randomly mutate in MN positions the last $(PS - Cross)$ elements of the matrix $[Poplist]$ by calling the *Mutation* function in Algorithm 4 and store the mutated elements into the (dummy) matrix *[Mutationlist]* on a per-row basis;
12: Compute and store the resource allocation vector and the corresponding consumed energy in (51) of each element of the matrices $[Childlist]$ and $[Mutationlist]$ through PS runs of the *RAP* solver in Algorithm 1;
13: Copy the first $Cross$ elements of the matrix $[Poplist]$ and the full matrices $[Childlist]$ and $[Mutationlist]$ into the (dummy) matrix $[Candidatelist]$;
14: Sort the $(PS + Cross)$ elements of the matrix $[Candidatelist]$ for increasing values of their consumed energy;
15: Copy the first PS elements of the matrix $[Candidatelist]$ into the matrix $[Poplist]$;
16: **if** energy of the first element of the matrix $[Poplist]$ is lower than the current value of $\mathcal{E}_{BEST}$ **then**
17: Copy the first element (i.e., the first row) of the matrix $[Poplist]$ into $\vec{x}_{BEST}$, $\overrightarrow{RS}_{BEST}$ and $\mathcal{E}_{BEST}$;
18: **end if**
19: **end for**
20: **return** $\vec{x}_{BEST}$, $\overrightarrow{RS}_{BEST}$ and $\mathcal{E}_{BEST}$.

Algorithm 3 Pseudo-Code of the Implemented Crossover Function

Input: Two allocation vectors: $\overrightarrow{Parent_1}$ and $\overrightarrow{Parent_2}$ to be crossed over.
Output: Two crossed-over allocation vectors $\overrightarrow{Child_1}$ and $\overrightarrow{Child_2}$.

▷ Initialization phase

1: Generate a random integer I over the interval $[2, V - 1]$;

▷ Perform the swapping operation

2: Copy the first I elements of $\overrightarrow{Parent_1}$ and $\overrightarrow{Parent_2}$ into the first I positions of $\overrightarrow{Child_1}$ and $\overrightarrow{Child_2}$, respectively;
3: Copy the last $(V - I)$ elements of $\overrightarrow{Parent_1}$ and $\overrightarrow{Parent_2}$ into the last $(V - I)$ positions of $\overrightarrow{Child_1}$ and $\overrightarrow{Child_2}$, respectively;
4: **return** $\overrightarrow{Child_1}$ and $\overrightarrow{Child_2}$.

Shortly, the implemented *Crossover* function: (i) generates a random pointer to the location index at which the crossover is performed (see step 1 of Algorithm 3); and, then, (ii) carries out the corresponding swapping of the task allocation input vectors: $\overrightarrow{Parent_1}$ and $\overrightarrow{Parent_2}$ (see steps 2 and 3 of Algorithm 3), so to return the crossed-over task allocation output vectors: $\overrightarrow{Child_1}$ and $\overrightarrow{Child_2}$ (see step 4 of Algorithm 3). The underlying rationale is to allow the crossed over output vectors to (hopefully) inherit the "good" fitness properties retained by the corresponding input vectors.

Algorithm 4 Pseudo-Code of the Implemented Mutation Function

Input: Task allocation vector $\vec{x}$ to be mutated;
Number MN of the scalar elements to be mutated.
Output: The mutated task allocation vector $\overrightarrow{mx}$.

▷ Initialization phase

1: Generate an MN-tuple random vector: $\vec{\ell}$ that points the positions to be mutated. Each element of $\vec{\ell}$ is an integer number over
the interval $[2,\ V-1]$;
2: Generate an MN-tuple random vector: $\overrightarrow{offset}$ that stores the mutated values. Each scalar element of $\overrightarrow{offset}$ takes value over the discrete set $\mathcal{A}$ in (2);
3: Copy $\vec{x}$ into $\overrightarrow{mx}$.

▷ Perform the mutations

4: **for** $j = 1 : MN$ **do**
5: Copy the j-th element of $\overrightarrow{offset}$ into the position of $\overrightarrow{mx}$ that is pointed by the j-th element of $\vec{\ell}$;
6: **end for**
7: **return** $\overrightarrow{mx}$.

The opposite goal is, indeed, pursued by the *Mutation* function of Algorithm 4. In fact, after generating a random vector of pointers to the locations to be mutated (see step 1 of Algorithm 4) and a random vector of mutated values (see step 2 of Algorithm 4), this function copies the generated mutated values at the pointed locations of the input vector $\vec{x}$ (see steps 4-6 of Algorithm 4). Afterwards, it returns the mutated output vector $\overrightarrow{mx}$ (see step 7 of Algorithm 4). Hence, the underlying rationale is to attempt to improve (if possible) the "bad" fitness of the input vector $\vec{x}$ by randomly changing a number MN of its randomly selected components (see step 4 of Algorithm 4).

B. PECULIAR FEATURES OF THE PROPOSED A-GTA-S AND ITS COMPUTATIONAL COMPLEXITY

An examination of the reported Algorithms 2, 3 and 4 unveils the following three main *peculiar* features of the proposed *A-GTA-S*.

First, since it leverages the *RAP* solution for the evaluation of the fitness in (70) of each candidate task allocation vector (see steps 3 and 12 of Algorithm 2), the proposed *A-GTA-S* inherits, by design, the *adaptive capability* natively retained by the *RAP* solution (see Section VII-B and related remarks). This feature makes its utilization appealing in the mobile scenario of Fig. 1, where both the throughput of the wireless connections and the computing capabilities of the discovered server nodes may undergo abrupt and unpredictable changes.

Second, formally speaking, we cannot claim that, in general, the solution in (71) returned by the A-GTA-S coincides with the optimal one in (53) of the *TAP*. However, the pursued elitary approach guarantees that the fitness of the solution returned by the *A-GTA-S* does not decrease for increasing values of the product: $PS \times G_{MAX}$ (see steps $16-18$ of Algorithm 2). This formal property assures, in turn, that the *A-GTA-S* solution in (71) asymptotically approaches the optimal one for growing value of the product: $PS \times G_{MAX}$.

Third, it has been experienced that elitary GAs may be too conservative, i.e., their solutions may be trapped by local minima at finite values of the product: $PS \times G_{MAX}$ (see, for example, Chapter 8 of [46] and references therein). Hence, in order to effectively cope with this potential drawback, the proposed *Mutation* function of Algorithm 4 randomizes *both* the locations of the elements to be mutated *and* the corresponding mutated values (see steps 1, 2 and 5 of Algorithm 4). In this regard, we have numerically experienced that values of the number MN of the mutated element in step 4 of Algorithm 4 of the order of about $V/2$ maximize the chances of the overall *A-GTA-S* escaping local minima and reach global (or, at least, quasi global) minima (see Section X). So doing, we anticipate that the simulation results of Section X support the conclusion that the proposed *A-GTA-S* is capable of attaining quasi-optimal energy performance at a reasonable low computational complexity. In this regard, an examination of Algorithm 2 points out that:

i. the solution of the *RAP* is invocated $((G_{MAX}+1)\times PS)$ times (see steps 3 and 12 of Algorithm 2). Furthermore, the computational cost of each *RAP* invocation is given by (69); and,
ii. $(G_{MAX}+1)$ sorting operations are carried out over sets composed by $(PS+Cross)$ elements (see step 14 of Algorithm 2). Furthermore, the computational cost of each sorting operation is of the order: $\mathcal{O}\left((PS+Cross)\times\log_2(PS+Cross)\right)$.

As a consequence, the overall computational complexity of Algorithm 2 is of the order:

$$\mathcal{O}\left((G_{MAX}+1)\times PS\times\left((3Q+5)\times I_{MAX}\right)+(PS+Cross)\times\log_2(PS+Cross)\right)\right), \quad (72)$$

and, then, it scales as:

$$\mathcal{O}\left(PS\times G_{MAX}\times(3Q+5)\times I_{MAX}\right), \quad (73)$$

for large values of the product: $G_{MAX}\times I_{MAX}\times PS$. We anticipate that, in Section X, the formula in (73) is exploited, in order to investigate about the right trade-off between the

two contrasting requirements of quasi-optimal energy performance and low implementation complexity of the proposed *A-GTA-S*.

IX. EcoMobiFog—THE PROPOSED TECHNOLOGICAL PLATFORM

The goal of this section is to sketch the main building blocks, offered services and control flows of *EcoMobiFog*, i.e., the proposed networked computer architecture for the actual support of the developed *JOP* solution of Sections VII and VIII.

Toward this end, we begin to note that, in the ecosystem of Fig. 1, the Mobile device queries the connected Cloud and/or Fog nodes for additional computing resources, while the last may cooperate through data exchange. This means, in turn, that, in our framework:

i. the Mobile-to-Fog and Mobile-to-Cloud interactions are of *Client-Server* type, with the Mobile device (resp., the Cloud and Fog nodes) that plays (resp., play) the role of Client (resp., Servers); and,
ii. the Fog-to-Fog and Fog-to-Cloud interactions needed for cooperative task executions are of *Peer-to-Peer* type.

According to this observation, the proposed *EcoMobiFog* technological platform for the support of the developed task offloading framework is composed of two main parts, i.e., a Mobile client part and a Cloud/Fog server part. Their main building blocks and exchanged control flows are sketched in Fig. 3. Specifically, according to Fig. 3, *EcoMobiFog* relies on *six* main agents that support the instantiated containers, namely, *Profilers*, *Task Managers*, *Connection Managers and Failure Handlers*, *Solvers*, *Controllers* and *Control Flows*. In the sequel, we describe the supported services, as well as their mutual interactions.

1) PROFILERS

Each container hosted by the Mobile, Cloud and Fog nodes is equipped with the corresponding Profiler. The function of the Profiler is to provide context-awareness for the associated container by performing in real-time the measurements of a number of context parameters, so to assist the *Controller* when needed. For this purpose, each Profiler is, in turn, composed of an *Application Profiler*, a *Processor Profiler* and a *Network Profiler*.

The *Application Profiler* tracks the execution state of the tasks processed by the underlying Container Engine of Fig. 3 by monitoring: the current set of tasks under execution, the related execution times $T_{i,N}^{EXE}$, and the sizes d_{ij} of the input data required for the task executions (see Sections III and IV).

The *Processor Profiler* works on a per-virtual processor basis, and (when needed) communicates the performed measurements to the *Controller*. In our framework, these measurements include: the per- virtual processor computing frequency f_N, its corresponding maximum value $f_N^{(MAX)}$, and the currently consumed computing energy $\mathcal{E}_N$ (see Section V-B).

The *Network Profiler* collects the network state of the connections currently sustained by the active NICs, in order to detect in real-time possible network changes. Specifically, jobs of a Network Profiler is to measure the currently available set of maximum throughput:$\{R_{N_1\rightarrow N_2}^{(MAX)}\}$ of the managed TCP/IP connections, as well as the actual parameters: $\{\Omega^{(Tx)}, \Omega^{(Rx)}, \xi^{(Tx)}, \xi^{(Rx)}\}$ of the corresponding per-connection power profiles (see Section V-C). From time to time, it returns to the *Controller* the set $\{\mathcal{E}_{N_1\rightarrow N_2}\}$ of the energy actually consumed by the managed connections.

2) TASK MANAGERS

Each container deployed at the Mobile, Cloud and Fog nodes is equipped with the corresponding Task Manager. Its main function is to implement the adopted Task Service Discipline (like, for example, the *SEQ* or *WPS* ones of Section IV-A) and, then, guarantee that the allocated tasks are executed in a compliant way.

3) CONNECTION MANAGERS AND FAILURE HANDLERS

Each container deployed at the Mobile, Cloud and Fog nodes is equipped with the corresponding *Connection Manager* and *Failure Handler* module. Its goal is to manage the operations of the multiple NICs that equip the hosting node. For this purpose, the *Failure Handler* module: (i) manages the time out-induced re-transmissions of the lost TCP segments; (ii) detects connection failure events and alerts the associated Network Profiler; and, (iii) measures in real-time the set: $\{\overline{NF}_{N_1\rightarrow N_2}\}$ of the failure rates of the ongoing connections (see Section V-C). In a parallel way, the *Connection Manager* performs: (i) *setup and tear-down* of the engaged TCP/IP connections; and, (ii) *node discovery*, i.e., it monitors the strength of the signal received by each NIC, in order to discover the presence of proximate nodes. About this last functionality, we further specify that, after discovering a new node, the Connection Manager updates the information concerning the discovered node (like, connection bandwidth, IP address, computing capacity and similar), and communicates the acquired information to the Profiler. However, since node discovery through NIC monitoring can potentially increase the resulting network energy consumption, in our framework, we plan that this operation is carried out from time to time, for example, on a periodical or event-driven basis.

4) SOLVERS

The goal of the *Solver* module of Fig. 3 is to implement the developed numerical procedure for the real-time evaluation of the solution of the *JOP* (see Sections VII and VIII). However, since the Mobile device is resource limited and its implemented functionalities must be held at the minimum, in the proposed framework, *only* the containers at the server nodes (i.e., at the Fog and Cloud nodes) are equipped with the Solver module and, then, *only* these containers are capable of computing the *JOP* solution in real-time. This means that, when the Controller at the mobile device decides to launch a task offloading procedure, it connects to an available Fog or Cloud node (thereinafter referred to as the *solving server*), and, then,

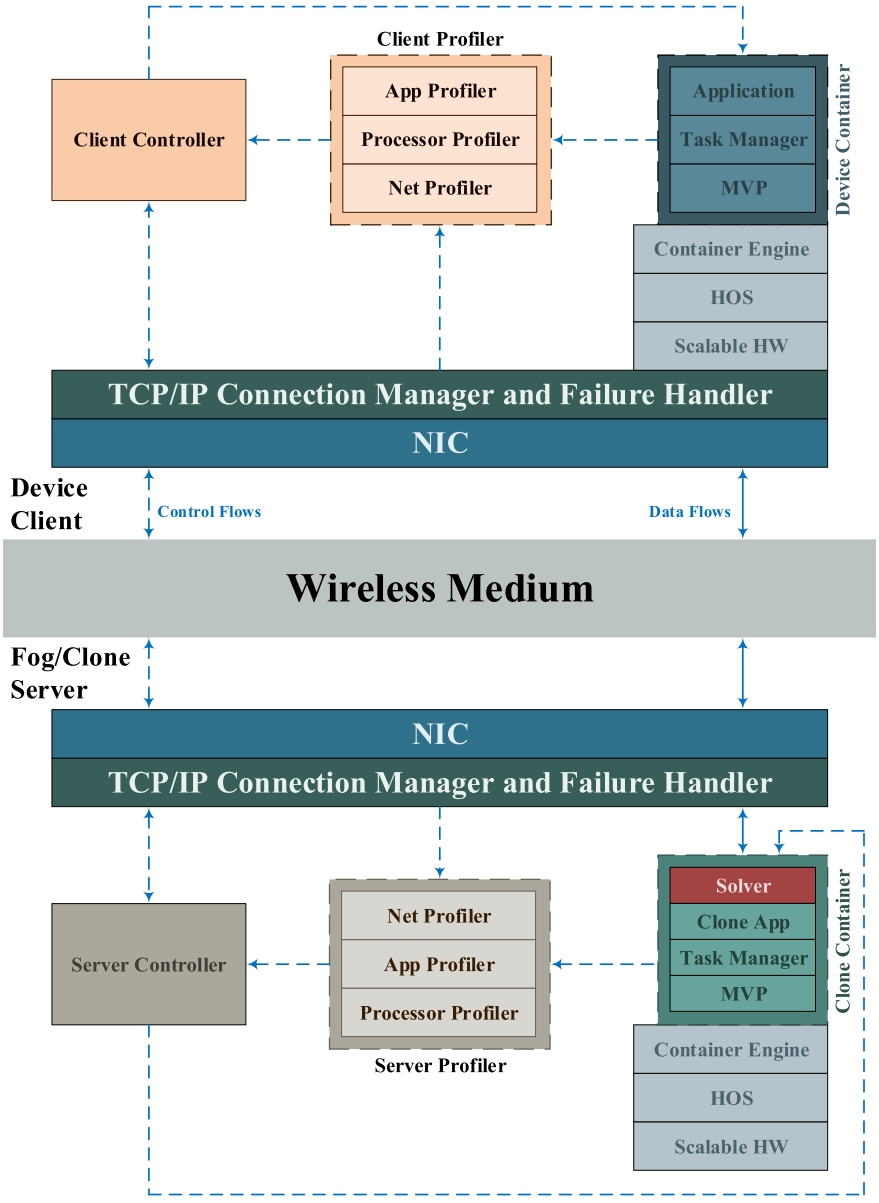

FIGURE 3. The proposed *EcoMobiFog* technological platform for the support of the developed *JOP* solution. Continuous (resp., dotted) arrowed paths denote data (resp., control) TCP/IP flows.

queries it to solve the underlying *JOP*. After computing the *JOP* solution, the solving server returns to both the Mobile device and all the other involved server nodes the computed solution in (49). Afterward, the involved Controllers self-synchronize, and, then, start the distributed execution of the underlying application.

5) CONTROLLERS

Each container hosted by the Mobile, Cloud and Fog nodes is equipped with a corresponding *Controller*. In our framework, it fulfils the following three main functions. First, it performs *intra-node* synchronization, i.e., it disciplines the actions of all other modules hosted by the node. Second, on the basis of the information received by the associated Profiler and Solver modules, it decides whether and where to offload the application tasks, and, then, dispatches the tasks to the appropriate nodes. Third, it performs *inter-node* synchronization, i.e., it cooperates with the Controllers hosted by the other nodes that are involved in the DAG execution, in order to guarantee the synchronized execution of the adopted inter-node Task Scheduling discipline (see the *STS* and *PTS* disciplines of Section IV-C).

6) CONTROL FLOWS

The aforementioned inter-node synchronization is supported by a set of end-to-end control flows, that connect (in a peer-to-peer fashion) all the Mobile, Cloud and Fog nodes of

the ecosystem of Fig. 1. Since these flows must be reliable, we assume that they are sustained by TCP/IP connections (see the dotted arrowed paths of Fig. 3). However, in our framework, all the server nodes are already equipped with a copy of the DAG to be executed (see the assumptions of Section III). As a consequence, the exchanged control data are reduce to: (i) the information data about the solution in (49) of the *JOP* that has been computed by the solving server; and, (ii) the barrier-based synchronization signaling, in order to guarantee that the distributed execution of the application DAG is compliant with the adopted inter-node Task Scheduling discipline. On the basis of these design choices, it is expected that the aggregate throughput of all involved control flows remains limited and does not significantly interfere with the throughput of the corresponding data flows (see the continuous arrowed paths of Fig. 3).

X. PERFORMANCE TESTS AND COMPARISONS

The goal of this section is to numerically check and compare the adaptive capability and energy-vs.-delay performance of the proposed *A-GTA-S* under multiple test operating scenarios. In order to suitably present the related multi-facet aspects, we organized this section according to the following roadmap. After describing in Section X-A the simulated environment and the considered benchmark strategies, the adaptive capability of the proposed *A-GTA-S* is checked in Section X-B, while Section X-C investigates its performance sensitivity on the population size, number of performed generations and fraction of the crossed over population. Sections X-D and X-E compare the resource/task allocations and energy consumption of the proposed *A-GTA-S* with respect to the corresponding ones of the considered benchmark strategies, while Section X-F checks the corresponding performance sensitivity on the computing-to-communication ratios of the considered benchmark DAGs. The goal of Section X-G is to check and compare the energy-vs.-delay performance of the proposed *A-GTA-S* under the Eco and Mobile-centric service models in (38) and (39). Finally, Section X-H carries out comparative tests of the *A-GTA-S* average energy performance when, due to the device mobility, the availability of the underlying Mobile-Fog connections undergoes random ON-OFF variations.

A. SIMULATED PLATFORM AND PURSUED COMPARISON METHODOLOGY

The most part of the numerical results and performance comparisons available in the open literature refers to three-tier Mobile-Fog-Cloud offloading systems, in which single Fog nodes are involved (see, for example, [11] and references therein). Hence, in order to present comparative simulation results, aligned with the current literature, we have simulated the Mobile-Fog-Cloud platform sketched in Fig. 4. From a formal point of view, it is the instance of the general ecosystem of Fig. 1 that is obtained by setting $Q = 1$ in (2).

Regarding the simulated platform of Fig. 4, the following two main introductory remarks are in order.

First, unless otherwise stated, in the sequel, it is understood that: (i) the settings of the main involved system parameters are the ones listed in the last column of final Table 11, where the communication technology sustaining the cellular Mobile-Cloud (resp., the short-range Mobile-Fog) connection is the 4G-LTE one [40] (resp., the IEEE 802.11b WiFi one [47]); (ii) the reported simulated results refer to the Eco-centric model of (38) under the *SEQ* and *STS* service task scheduling disciplines of (9) and (18), respectively; and, (iii) the simulated maximum allowed per-DAG execution time $T_{DAG}^{(MAX)}$ in (37b) ranges over the interval: $0.3 - 2.4$ (s).

Second, the performed simulations have been carried out by using the recent *VirtFogSim* toolbox [48], atop a hardware execution platform equipped with: (i) an Intel 10-core i9-7900X processor; (ii) a GPU ZOTAC GetForce GTX 1070; (iii) an SSD with 512 GB plus an HHD with 2 TB; and, (iv) 32 GB of RAM DDR 4. Furthermore, the simulation code exploits the release R2018a of MATLAB as software execution environment.

1) PURSUED COMPARISON METHODOLOGY AND BENCHMARK STRATEGIES

Since the main peculiar features of the proposed *A-GTA-S* solving approach is that it affords the two related problems of the adaptive resource *and* task allocations in a *joint* way. It is reasonable to pursue a methodology in which three main families of benchmark strategies are considered for comparison purpose.

Specifically, a first benchmark family of published strategies focuses on the optimization of the task allocation under *fixed* (i.e., *not* optimized) allocation of the underlying computing and/or networking resources. For this purpose, various versions of more or less optimized heuristic and metaheuristic task allocation algorithms have been proposed as building blocks of Middleware technological platforms, like, for example, *MAUI* [21], *CloneCloud* [49], *mCloud* [27], and *StreamCloud* [26], just to name but a few. In a nutshell, all these solutions focus on the constrained minimization of the task execution times (or the consumed energy) under *fixed* resource allocation vectors.

A second family of published offloading strategies affords the dual problem of the constrained optimal allocation of the computing and/or networking resources under *fixed* (i.e., not optimized) task offloading. These strategies mainly refer to single/multi-user two-tier MEC environments, and they are reviewed, for, example, in [12, Section 3].

A last family of contributions aim at pursuing (when it is possible) an optimal solving approach that directly exploits the *Exhaustive Search (ES)* [12]. Taking this approach, the *full* space of the allowed task allocation vectors is thoroughly searched for the global best task and optimization of the allocated resources is also performed under each searched task allocation vector. Although it is guaranteed that the returned solution is the optimal one, the computational complexities of these ES-based solving approaches scale in an *exponential* way with the size V of the underlying application

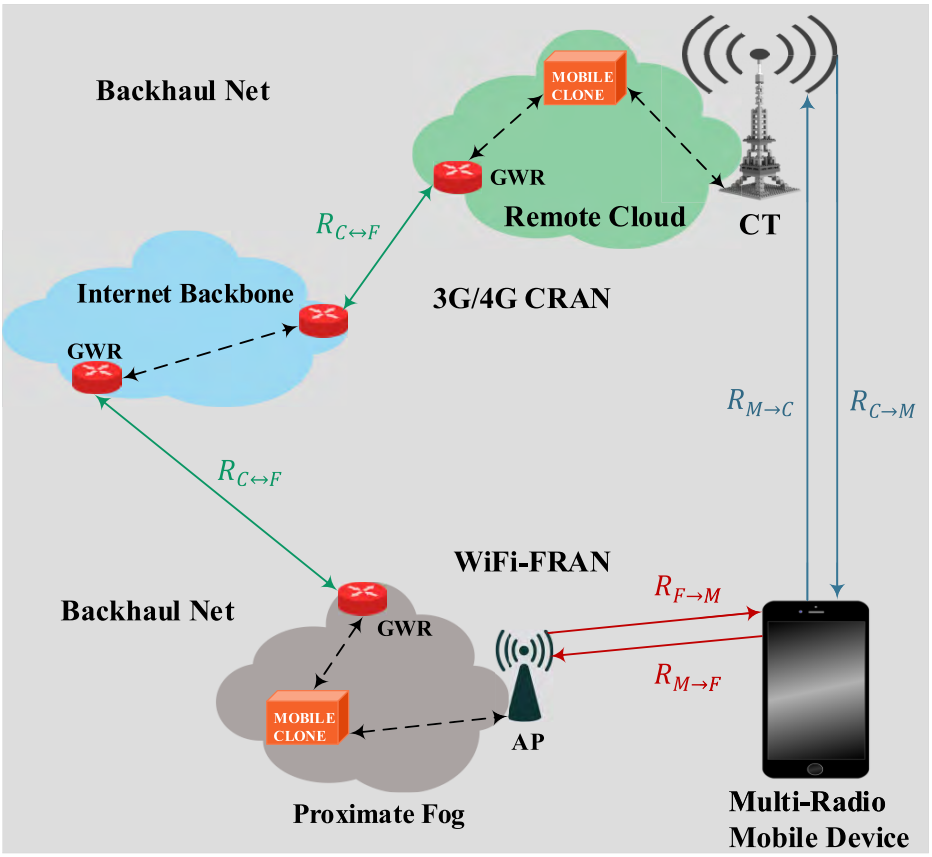

FIGURE 4. The simulated three-tier Mobile-Fog-Cloud platform.

DAGs, so that these approaches are typically applied to "toy" examples.

Overall, on the basis of these considerations, in the sequel, the performance of the proposed *A-GTA-S* will be compared with the corresponding ones of the following five benchmark strategies:

1. *Only Task Allocation Strategy (OTA-S)*: by definition, the *OTA-S* runs the GA of Algorithm 2 under the fixed (i.e., not optimized and time-invariant) maximal resource allocation vector in (58). This means, in turn, that in steps 3 and 12 of Algorithm 2, *OTA-S* evaluates the energy consumed by the involved task allocation vectors by simply setting the required resource allocation vector $\overrightarrow{RS}$ at the maximal one $\overrightarrow{RS}^{MAX}$. In the sequel, the energy consumed under *O-TAS* is indicated as $\mathcal{E}_{OTA-S}$. *O-TAS* is representative of the (aforementioned) first family of state-of-the-art offloading strategies;
2. *Adaptive Only Fog Strategy (A-OF-S)*: *A-OF-S* assumes that the first and last tasks of the underlying DAG are executed by the Mobile device, while all the other tasks are executed by the Fog node of Fig. 4. Hence, by definition, *A-OF-S* fixes the task allocation vector $\vec{x}$ as in:
$$\vec{x} \equiv \vec{x}_{FOG} \stackrel{\text{def}}{=} [M, F, \ldots, F, M], \tag{74}$$
and, then, invokes the *RAP* solution of Algorithm 1, in order to compute the optimal resource allocation vector $\overrightarrow{RS}_{A-OF-S}$ and the resulting consumed energy $\mathcal{E}_{A-OF-S}$ under $\vec{x}_{FOG}$. *A-OF-S* falls into the second family of offloading strategy;
3. *Adaptive Only Cloud Strategy (A-OC-S)*: *A-OC-S* assumes that the first and last tasks of the underlying DAG are executed by the Mobile device, while all the other tasks are executed by the Cloud node of Fig. 4. Hence, by definition, *A-OC-S* fixes the task allocation vector $\vec{x}$ as in:
$$\vec{x} \equiv \vec{x}_{CLD} \stackrel{\text{def}}{=} [M, C, \ldots, C, M], \tag{75}$$
and, then, invokes the *RAP* solution of Algorithm 1, in order to compute the optimal resource allocation vector $\overrightarrow{RS}_{A-OC-S}$ and the resulting consumed energy $\mathcal{E}_{A-OC-S}$ under $\vec{x}_{CLD}$. *A-OC-S* is an instance of the second family of offloading strategy;
4. *Adaptive Only Mobile Strategy (A-OM-S)*: *A-OM-S* assumes that all tasks are executed (when it is feasible) by the Mobile device of Fig. 4. Hence, by definition, *A-OM-S* puts:
$$\vec{x} \equiv \vec{x}_{MOB} \stackrel{\text{def}}{=} [M, M, \ldots, M, M], \tag{76}$$
and, then, invokes the *RAP* solution of Algorithm 1, in order to compute the optimal resource allocation vector $\overrightarrow{RS}_{A-OM-S}$ and the resulting consumed energy $\mathcal{E}_{A-OM-S}$

under $\vec{x}_{MOB}$. The *A-OM-S* may be considered as a (limit) instance of the second family of task allocation strategies;

5. *Adaptive Exhaustive Search Strategy (A-ES-S)*: by design, *A-ES-S*: (i) generates all the $3^{(V-2)}$ task allocation vectors; (ii) evaluates the corresponding optimal resource allocation vectors and consumed energy through $3^{(V-2)}$ calls of the *RAP* solving procedure in Algorithm 1; and, finally: (iii) reports the task allocation vector $\vec{x}_{A-ES-S}$, and resource allocation vector: $\overrightarrow{RS}_{A-ES-S}$ that correspond to the minimum consumed energy: $\mathcal{E}_{A-ES-S}$. By design, these last coincide with the solution in (49) of the *JOP*, that is, it is guaranteed that:

$$\vec{x}_{A-ES-S} \equiv \vec{x}^*, \quad \overrightarrow{RS}_{A-ES-S} \equiv \overrightarrow{RS}^*, \quad \mathcal{E}_{A-ES-S} \equiv \mathcal{E}^*. \tag{77}$$

The computational complexities of the five considered benchmark strategies span a large range. Hence, in order to carry out fair performance-vs.-computational complexity comparisons, Table 3 reports these complexities, together with the corresponding one of the proposed *A-GTA-S* (see the last row of Table 3). The reported formulas account for the fact that, under the simulated scenario of Fig. 4, the computing complexity of the *RAP* scales as (see (69) with $Q = 1$): $\mathcal{O}(8 \times I_{MAX})$.

TABLE 3. Computational complexity of the simulated task offloading strategies; A := Adaptive.

Simulated Strategy	Asymptotic computational complexity
OTA-S	$\mathcal{O}(PS \times G_{MAX})$
A-OF-S	$\mathcal{O}(8 \times I_{MAX})$
A-OC-S	$\mathcal{O}(8 \times I_{MAX})$
A-OM-S	$\mathcal{O}(8 \times I_{MAX})$
A-ES-S	$\mathcal{O}(8 \times 3^{(V-2)})$
A-GTA-S	$\mathcal{O}(8 \times PS \times G_{MAX})$

2) A FIRST SET OF TEST DAGs

Fig. 5 sketches the topology of a first set of test DAGs. The rationale behind their considerations is that they exhibit the three basic topologies (i.e., the Mesh, Tree and Hybrid topologies) typically retained by DAGs for mobile stream applications [1], [26]. In this regard, we point out that: (i) in order to carry out fair performance comparisons, the summation of task workloads (resp., edge weights) of all DAGs of Fig. 5 are normalized to 3.32 (Mbit) (resp., 1.66 (Mbit)); and, (ii) a more complex (but, more application-specific) real-world DAG will be introduced in Fig. 15 and used for the final tests of Sections X-G and X-H.

B. TESTING THE ADAPTIVE CAPABILITY OF THE DEVELOPED RESOURCE ALLOCATION STRATEGY

The goal of this section is to test the sensitivity of the tracking capability of the *RAP* iterations in (65) and (66) on the clipping factor a_{MAX} of (67) and (68), as well as to evaluate the required convergence time I_{MAX} (in multiple of the iteration index m).

For this purpose, we have simulated a time-varying testing scenario in which, due to the mobility of the device of Fig. 4, both the Mobile-Fog up/down WiFi maximal throughput and the corresponding task allocation vectors undergo abrupt (and unpredicted) changes at the iteration indexes $m = 1$, 1000, 2000, 3000 and 4000. Specifically, in the simulated setting, we have that: (i) at $m = 1$, the up/down cellular (resp., WiFi) connections are turned ON (resp., turned OFF) and all tasks are allocated to the Cloud node, i.e. (see (75)), $\vec{x} \equiv \vec{x}_{CLD}$; (ii) at $m = 1000$, the up/down WiFi connections are turned ON and all tasks are allocated to the Mobile node, i.e. (see (76)), $\vec{x} \equiv \vec{x}_{MOB}$; (iii) at $m = 2000$, the WiFi connections are still turned OFF and all tasks are re-allocated to the Cloud node, i.e., $\vec{x} \equiv \vec{x}_{CLD}$; (iv) at $m = 3000$, the up/down WiFi connections are turned ON once time, and all tasks are allocated to the Fog node, i.e. (see (74)), $\vec{x} \equiv \vec{x}_{FOG}$; and, finally, (v) at $m = 4000$, the up/down WiFi connections are definitively turned OFF and all tasks are re-migrated to the Cloud node, i.e., $\vec{x} \equiv \vec{x}_{CLD}$. After each change of the setup environment, the *RAP* solution is computed by running the iterations in (65) and (66), in order to properly re-allocate both the per-clone computing frequencies at the Mobile-Fog-Cloud nodes and the corresponding up/down Cellular-WiFi throughput.

The obtained dynamic behaviors of the total consumed energy $\widetilde{\mathcal{E}}_{TOT}$, network energy $\widetilde{\mathcal{E}}_{NET}$ and *lambda* multipliers $\tilde{\lambda}$ returned by the *RAP* solution are reported in Figs. 6, 7 and 8 for three test values of the clipping factor a_{MAX} and under *DAG1*, *DAG2* and *DAG3*, respectively.

An examination of the reported time-plots leads to three main insights. First, even in the presence of the (aforementioned) abrupt changes of the environmental setup, the corresponding *lambda* multipliers remain almost surely vanishing (see the bottom parts of Figs. 6, 7 and 8). This supports the conclusion that all the resource allocations computed by the RAP solution are, indeed, *feasible* (i.e., they meet the *RAP* feasibility constraint in (59)). Second, the abrupt step-like jumps of the plots of $\widetilde{\mathcal{E}}_{NET}$ in the middle parts of Figs. 6, 7 and 8 are due to the combined effects of *both* the changes of the availability of the WiFi connection and the re-allocation of the Cellular up/down throughput triggered by the underlying execution of the *RAP* iterations. Third, a comparative examination of the red-green-blue colored plots of the upper parts in Figs. 6, 7, and 8 confirms that bigger values of the clipping factor a_{MAX} of (67) and (68) speed up the convergence to the corresponding steady-states, but also tend to introduce larger steady-state oscillations.

Overall, two final insights stem from the carried out tracking analysis. First, at least in the carried out tests, values of a_{MAX} ranging over the interval: 1.0×10^{-7} — 2.5×10^{-7} guarantee good trade-offs among the contrasting requirements of quick reaction to mobility-induced variations of the operative environment and stable behavior in the steady-state.

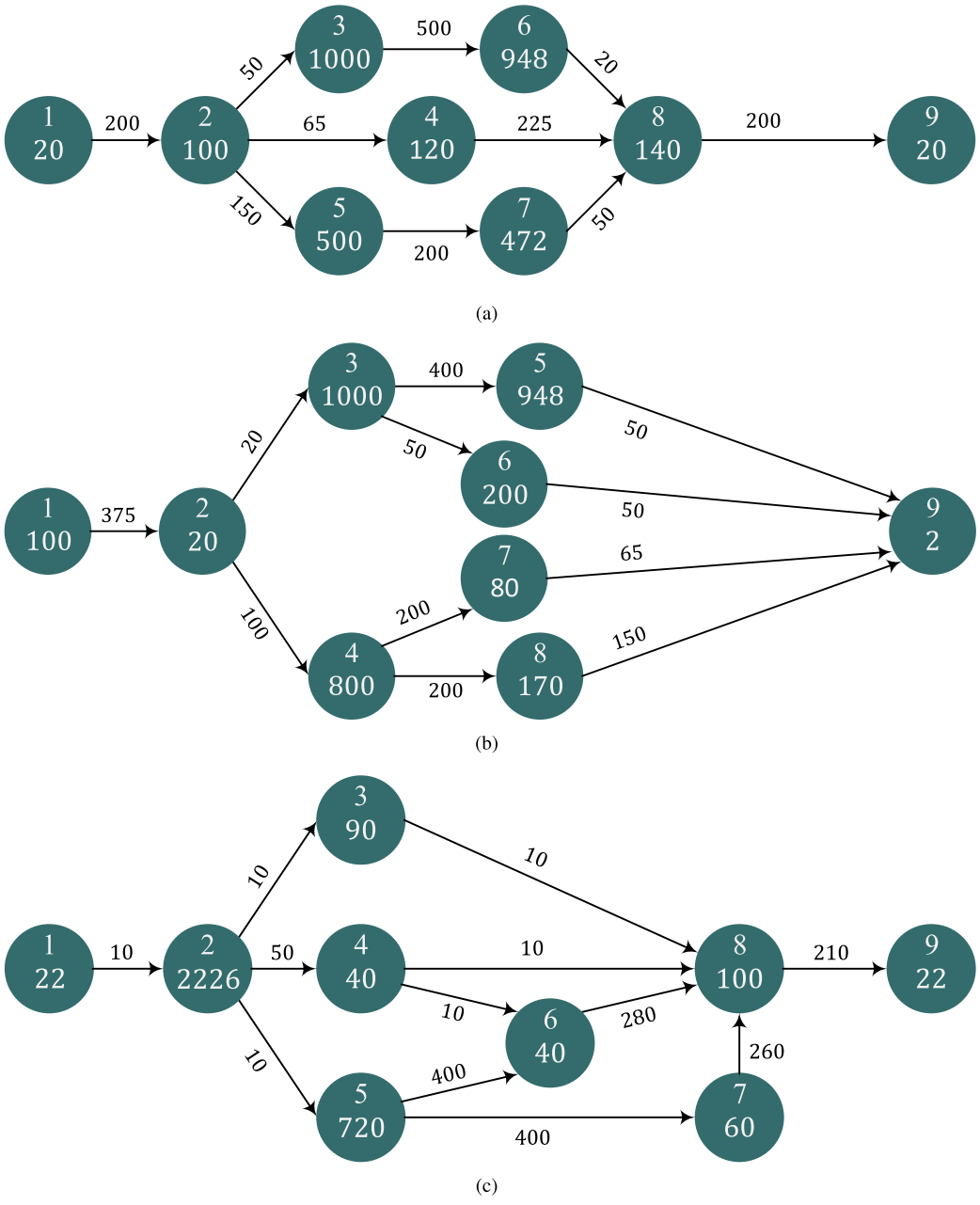

FIGURE 5. A first set of test DAGs: (a) DAG1: Mesh topology; (b) DAG2: Tree topology; and, (c) DAG3: Hybrid topology. Task workloads and edge weights are in (kbit). In all cases, the summations of the task workloads and edge weights equate to 3.32 (Mbit) and 1.66 (Mbit), respectively.

Second, a number of the primal-dual iterations I_{MAX} limited up to 450 — 600 suffices, in order to reach stable resource allocations in the presence of abrupt environmental changes.

C. TUNING THE ENERGY PERFORMANCE-VS.-COMPUTATIONAL COMPLEXITY TRADE-OFF OF THE PROPOSED A-GTA STRATEGY

The goal of this section is to check the sensitivity of the energy performance of the proposed *A-GTA-S* on the input parameters PS, CF and G_{MAX} of Algorithm 2, that specify the population size, fraction of crossed over population and number of performed generations of the underlying GA. The obtained numerical results are reported by the bar plots of Figs. 9, 10 and 11 under *DAG1*, *DAG2*, and *DAG3*, respectively. In order to put the reported results under a right perspective, we note that: (i) since we have numerically ascertained that the obtained energy performance mainly depends on the product population size by generation number: $PS \times G_{MAX}$, the reported plots refer to the cases of $PS = 4, 8, 12, 16$, and 20 at fixed $G_{MAX} = 10$;

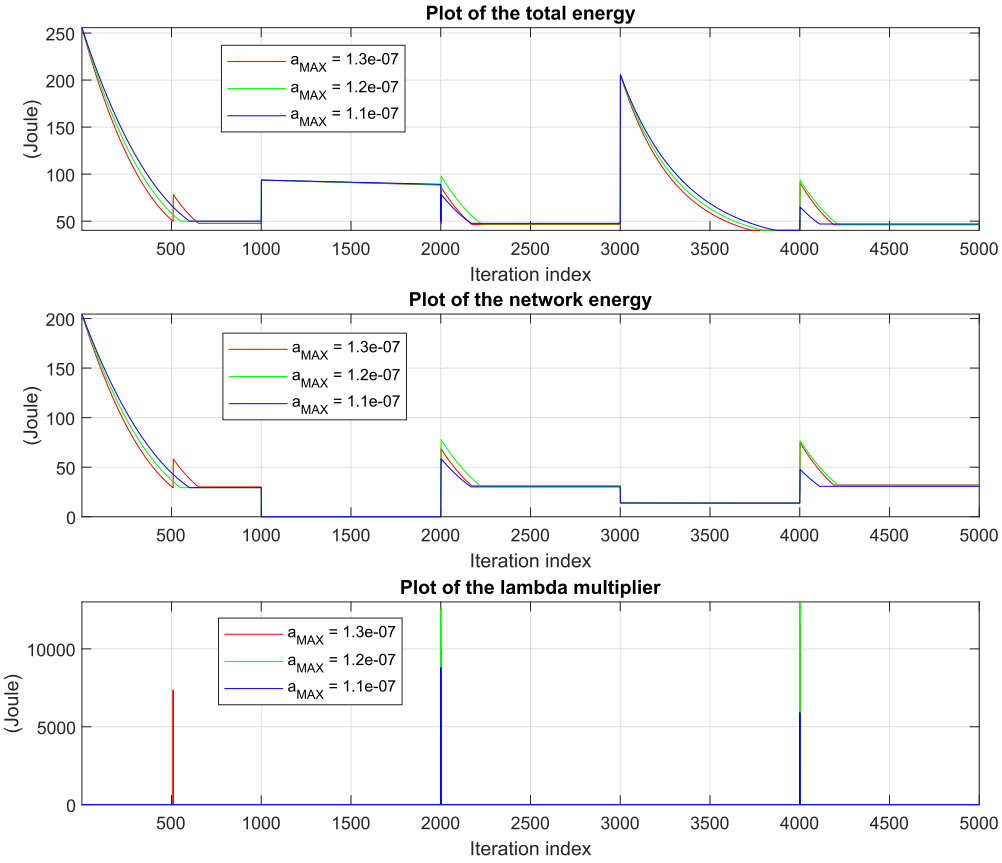

FIGURE 6. Tracking capability of the developed *RAP* function under DAG1 at $T_{DAG}^{(MAX)} = 0.3$ (s). (Top) Time behavior of $\tilde{\mathcal{E}}_{TOT}$; (middle) Time behavior of $\tilde{\mathcal{E}}_{NET}$; and, (bottom) Time behavior of $\tilde{\lambda}$ multiplier.

(ii) each bar plot reports the average, maximum and minimum energy consumption of the *A-GTA-S* over 50 independent trials; (iii) as ultimate benchmark, the corresponding energy $\mathcal{E}_{A-ES-S}$ returned by the *A-ES-S* are also reported in Figs. 9, 10 and 11.

A comparative examination of these bar plots lead to three main insights.

First, at fixed CF and for increasing values of PS, the average energy consumed by the proposed *A-GTA-S* monotonically decreases and reaches the benchmark ones of the corresponding *A-ES-S* at $PS = 20$. At the same time, the associated energy jitters decrease for increasing PS and tend to vanish at $PS = 20$. These monotonic behaviors are compliant with the (previously remarked) elitary nature of the implemented GA and support its actual effectiveness in the considered application scenarios.

Second, at a fixed PS, both the average energy and energy jitters of the A-GTA-S tend to increase for $CF < 0.5$ and $CF > 0.5$, while they tend to reach their respective minima at $CF \cong 0.5$.

Third, the above two trends are the same under all three test DAGs (i.e., they seem not to be so sensitive on the topologies of the considered DAGs) and we have numerically ascertained that they occur under the overall tested spectrum of values of $T_{DAG}^{(MAX)}$.

Overall, the final insight stemming from the analysis of the bar plots of Figs. 9, 10 and 11 is that, at least in the carried out tests, the setting: $PS = 20$ and $CF = 0.5$ is the most energy performing one. Under this setting, the ratio between the implementation complexities of the benchmark *E-ES-S* and the proposed *A-GTA-S* remains quite high and of the order of (see Table 3): $3^7/(20 \times 10) \cong 11$.

D. TASK AND RESOURCE ALLOCATION PERFORMANCE OF THE PROPOSED A-GTA-S

In this section, we test the task placement, resource allocation and energy performance of the proposed *A-GTA-S* under the considered test DAGs of Fig. 5. This is done for values of the maximum allowed execution time $T_{DAG}^{(MAX)}$ of 0.3, 0.6, 1.2, and 2.4(s). The final goal is to acquire insights about the effects of the DAG topology on the task allocation patterns and energy consumption featuring the proposed *A-GTA-S*. In this regard, three main sets of conclusions stem from the numerical results reported in Tables 4, 5 and 6.

1) SENSITIVITY OF THE A-GTA-S PERFORMANCE TO THE ALLOWED MAXIMUM EXECUTION TIMES

A comparative examination of the numerical values reported in the 13-th columns of Tables 4, 5, and 6 points out that,

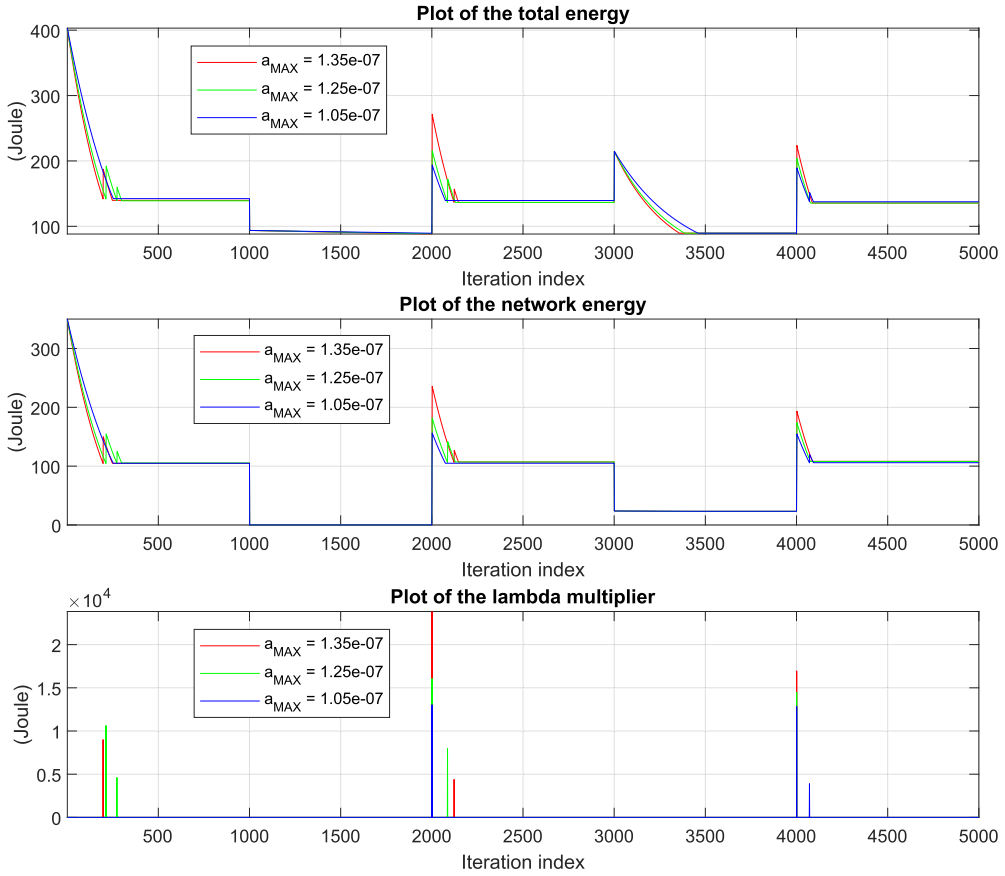

FIGURE 7. Tracking capability of the developed *RAP* solution under DAG2 at $T_{DAG}^{(MAX)} = 0.3$ (s). (Top) Time behavior of $\tilde{\mathcal{E}}_{TOT}$; (middle) Time behavior of $\tilde{\mathcal{E}}_{NET}$; and, (bottom) Time behavior of $\tilde{\lambda}$ multiplier.

in all simulated cases, the total energy $\mathcal{E}_{TOT}$ consumed by the proposed *A-GTA-S* remains limited up to 42 (Joule). Since all the test DAGs of Fig. 5 share the same sum values of the task workloads and edge weights, this support the conclusion that these factors play the major role in dictating the energy efficiency of the performed task and resource allocations. However, a more detailed examination of Table 4, 5 and 6 also unveils two interesting trends.

First, under a fixed DAG, the consumed energy tends to decrease for increasing values of $T_{DAG}^{(MAX)}$. Roughly speaking, this first trend is due to the fact that larger values of $T_{DAG}^{(MAX)}$ allow the *RAP* solution of Algorithm 1 to lower the steady-state computing frequencies and/or the corresponding per-connection throughput (see the numerical values reported by the corresponding columns of Tables 4, 5 and 6). This reduces, in turn, the dynamic (i.e., resource-depending) components of the total consumed energy (see the energy models of Section V).

The second trend arises from the observation that, in general, at a fixed $T_{DAG}^{(MAX)}$, the energy consumption returned by the proposed *A-GTA-S* tends to be larger under the tree topology of *DAG2*, and, then, it somewhat decreases under the mesh topology of *DAG1* and the hybrid topology of *DAG3*. Intuitively, this trend is caused by the behavior of the corresponding network energy $\mathcal{E}_{NET}$. In fact, a comparative examination of the results reported in the last columns of Tables 4, 5 and 6 points out that the network energy consumed by the tree topology are larger than the corresponding ones of the mesh and hybrid topologies. This behavior is, indeed, compliant with the observation that the tree topology offers, by design, the smallest number of input-output paths. This forces the *A-GTA-S* to increase, in turn, the data traffic allocated to each input-output path, so that the dynamic (i.e., throughput-depending) components of the per-connection network energy increase too (see (26) and (28)).

2) FEATURES OF THE TASK ALLOCATION PATTERNS RETURNED BY A-GTA-S

Columns 2, 3, and 4 of Tables 4, 5 and 6 report the ID numbers of the tasks allocated by the *A-GTA-S* to the Mobile, Fog and Cloud nodes under *DAG1*, *DAG2*, and *DAG3*, respectively. Although the reported allocation patterns may strongly depend on the specifically considered DAGs, two main trends may be detected. First, at low values of $T_{DAG}^{(MAX)}$, medium-size communication-intensive tasks are typically allocated to

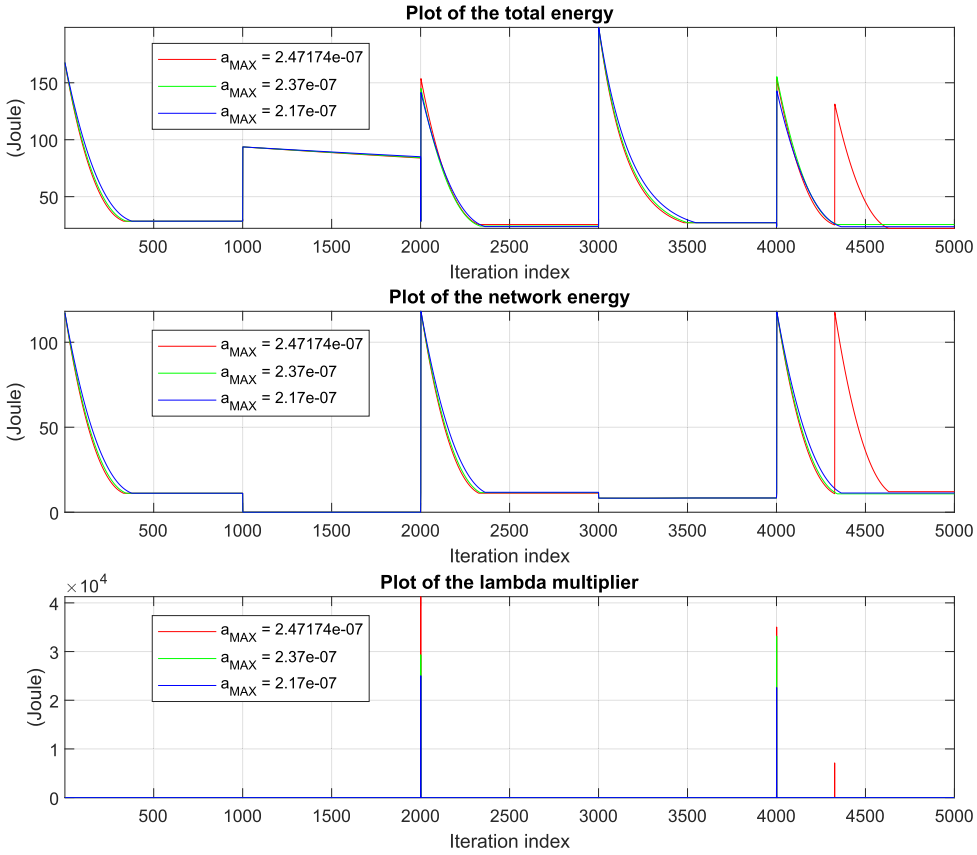

FIGURE 8. Tracking capability of the developed *RAP* function under DAG3 at $T_{DAG}^{(MAX)} = 0.3$ (s). (Top) Time behavior of $\tilde{\mathcal{E}}_{TOT}$; (middle) Time behavior of $\tilde{\mathcal{E}}_{NET}$; and, (bottom) Time behavior of $\tilde{\lambda}$ multiplier.

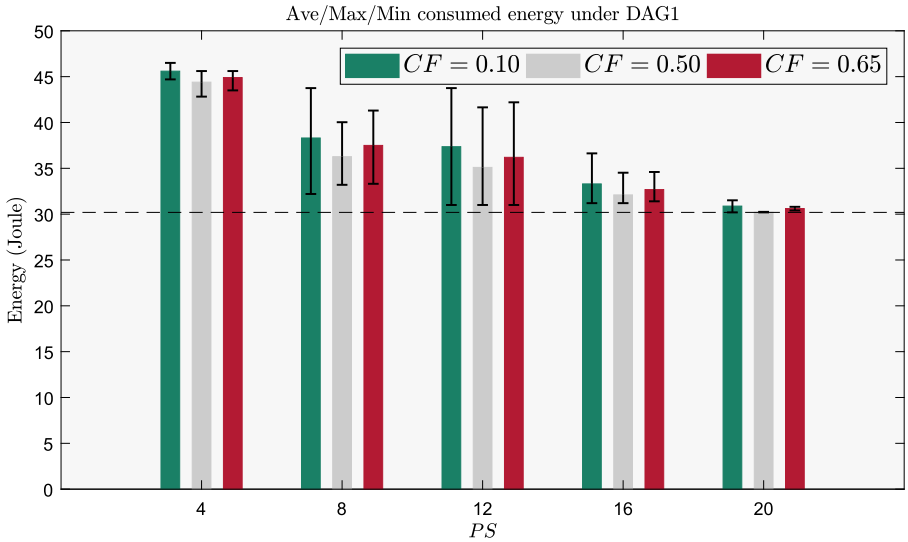

FIGURE 9. Bar plots of the average energy and energy jitters of the proposed *A-GTA* strategy for various values of the population size *PS* and crossover fraction *CF* under DAG1 at $T_{DAG}^{(MAX)} = 0.3$ (s), $G_{MAX} = 10$, and $MN = \text{round}\left((V-2)/2\right)$. As ultimate benchmark, the (horizontal) dashed line reports the corresponding energy consumed by the *A-ES* strategy.

the Fog node, while large-size communication-light tasks are assigned by *A-GTA-S* to the Cloud node. The Mobile device typically executes small-size communication-intensive tasks. Second, an increasing number of tasks are shifted from the Cloud node to the Fog node and/or to the Mobile device when $T_{DAG}^{(MAX)}$ decreases more and more.

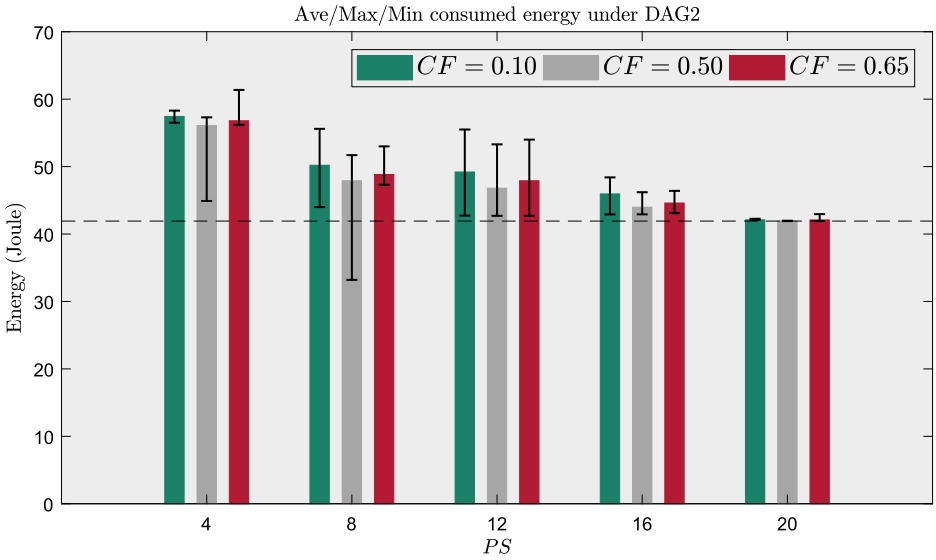

FIGURE 10. Bar plots of the average energy and energy jitters of the proposed *A-GTA* strategy for various values of the population size *PS* and crossover fraction *CF* under DAG2 at $T_{DAG}^{(MAX)} = 0.3$ (s), $G_{MAX} = 10$, and $MN = \text{round}\left((V-2)/2\right)$. As ultimate benchmark, the (horizontal) dashed line reports the corresponding energy consumed by the *A-ES* strategy.

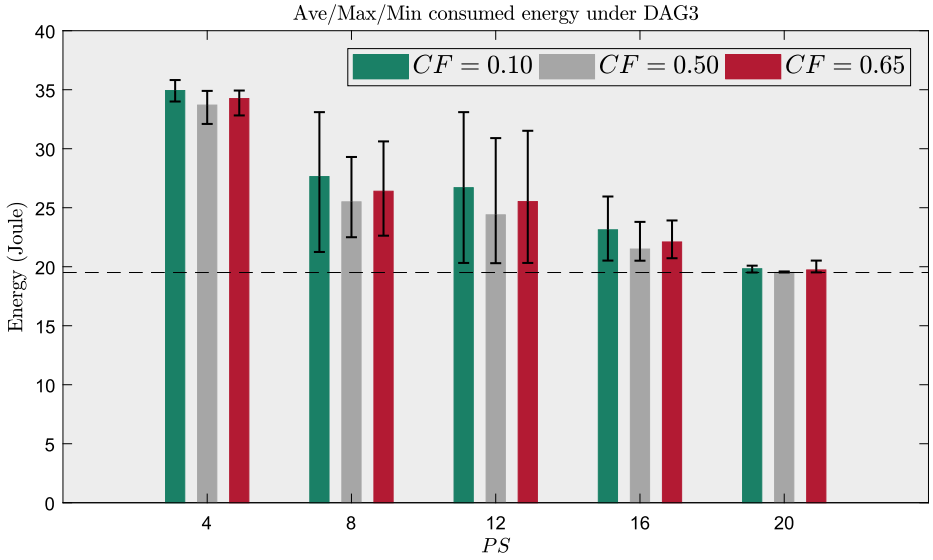

FIGURE 11. Bar plots of the average energy and energy jitters of the proposed *A-GTA* strategy for various values of the population size *PS* and crossover fraction *CF* under DAG3 at $T_{DAG}^{(MAX)} = 0.3$ (s), $G_{MAX} = 10$, and $MN = \text{round}\left((V-2)/2\right)$. As ultimate benchmark, the (horizontal) dashed line reports the corresponding energy consumed by the *A-ES* strategy.

3) HOW THE A-GTA-S EXPLOITS THE FOG-CLOUD BACKHAUL CONNECTION

A native feature of the three-tier platform of Fig. 4 is the presence of a (possibly, multi-hop and/or wired) two-way backhaul connection, that interconnects the Fog and Cloud nodes. Hence, it may be of interest to attain insight about how the proposed *A-GTA-S* exploits this auxiliary connection. Intuitively, we expect that the backhaul connection is utilized when there are large-size tasks to be allocated to the Cloud and the volumes of data output by the execution of these tasks are also large. Therefore, since the up/down Fog-Mobile WiFi connections of Fig. 4 are more energy efficient than the corresponding Cloud-Mobile cellular ones, it may be energy-saving to transport the processed data from/to the Cloud to/from the Fog over the two-way backhaul connection. So doing, the Fog node of Fig. 4 acts as *relay* node by forwarding the needed data over the WiFi up/down connections of Fig. 4. This is, indeed, the general strategy followed

TABLE 4. Task allocation, resource allocation and energy consumption of the proposed *A-GTA* strategy under *DAG1*. $T_{DAG}^{(MAX)}$ is measured in (s), all the resources are measured in (Mb/s) while the energy is measured in (Joule).

$T_{DAG}^{(MAX)}$	Mobile Tasks	Fog Tasks	Cloud Tasks	f_M	f_F	f_C	$f_{M\to F}$	$f_{F\to M}$	$f_{M\to C}$	$f_{C\to M}$	$f_{C\leftrightarrow F}$	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$
0.3	$\{1,9\}$	$\{2,4,5,7,8\}$	$\{3,6\}$	11.99	4.44	2.47	7.46	8.24	0.00	0.00	3.70	30.19	13.37
0.6	$\{1,9\}$	$\{2,4,8\}$	$\{3,5,6,7\}$	11.88	3.24	2.67	6.78	7.28	0.00	0.00	3.70	27.10	11.84
1.2	$\{1,9\}$	$\{2,4,8\}$	$\{3,5,6,7\}$	11.56	3.04	2.65	6.58	7.18	0.00	0.00	3.70	26.35	10.82
2.4	$\{1,8,9\}$	$\{2,4\}$	$\{3,5,6,7\}$	11.78	2.54	2.55	6.26	7.05	0.00	0.00	3.70	25.84	10.64

TABLE 5. Task allocation, resource allocation and energy consumption of the proposed *A-GTA* strategy under *DAG2*. $T_{DAG}^{(MAX)}$ is measured in (s), all the resources are measured in (Mb/s) while the energy is measured in (Joule).

$T_{DAG}^{(MAX)}$	Mobile Tasks	Fog Tasks	Cloud Tasks	f_M	f_F	f_C	$f_{M\to F}$	$f_{F\to M}$	$f_{M\to C}$	$f_{C\to M}$	$f_{C\leftrightarrow F}$	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$
0.3	$\{1,2,9\}$	$\{4,6,7,8\}$	$\{3,5\}$	11.97	3.76	2.97	7.77	8.16	4.12	1.45	3.70	41.92	20.47
0.6	$\{1,2,9\}$	$\{3,8\}$	$\{-\}$	11.92	1.85	0.00	7.01	5.96	0.00	0.00	0.00	28.60	11.12
1.2	$\{1,2,9\}$	$\{3,8\}$	$\{-\}$	11.79	1.83	0.00	6.95	5.87	0.00	0.00	0.00	27.30	10.81
2.4	$\{1,4,9\}$	$\{5,8\}$	$\{-\}$	11.96	1.54	0.00	6.75	5.65	0.00	0.00	0.00	27.03	10.62

TABLE 6. Task allocation, resource allocation and energy consumption of the proposed *A-GTA* strategy under *DAG3*. $T_{DAG}^{(MAX)}$ is measured in (s), all the resources are measured in (Mb/s) while the energy is measured in (Joule).

$T_{DAG}^{(MAX)}$	Mobile Tasks	Fog Tasks	Cloud Tasks	f_M	f_F	f_C	$f_{M\to F}$	$f_{F\to M}$	$f_{M\to C}$	$f_{C\to M}$	$f_{C\leftrightarrow F}$	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$
0.3	$\{1,9\}$	$\{5,8\}$	$\{2,4\}$	11.95	2.36	2.48	0.00	5.91	1.35	0.00	3.70	19.51	5.93
0.6	$\{1,9\}$	$\{5,8\}$	$\{2,4\}$	11.91	2.28	2.37	0.00	5.82	1.31	0.00	3.70	19.37	5.43
1.2	$\{1,9\}$	$\{5,8\}$	$\{2,3\}$	11.81	2.36	2.15	0.00	5.61	1.24	0.00	3.70	19.21	5.28
2.4	$\{1,2,9\}$	$\{4,8\}$	$\{3\}$	11.94	2.16	2.01	0.00	5.22	1.19	0.00	3.70	18.13	5.01

by the *A-GTA-S*, in order to allow energy-efficient executions of *DAG1* and *DAG3* under all the considered spectrum of allowed maximum DAG execution times $T_{DAG}^{(MAX)}$. In fact, an examination of the corresponding Tables 4 and 6 points out that the optimized execution strategy returned by *A-GTA-S* utilizes: (i) the Cellular/Wifi up-connections of Fig. 4 for uploading the data to be processed by the Cloud/Fog nodes; (ii) the two-way backhaul connection: $C \leftrightarrow F$, in order to allow the Cloud and Fog nodes to exchange partially processed data; and, (iii) the WiFi down connection: $F \to M$ for the final delivering of the processed data to the Mobile device.

The (somewhat unexpected) final lesson is that, at least in the described operating scenarios, the utilization of (single-hop) Cloud-Mobile and/or Mobil-Fog links are *less* energy-efficient than the exploitation of the (multi-hop) Cloud-Fog-Mobile path.

E. PERFORMANCE COMPARISONS AGAINST THE BENCHMARK STRATEGIES

In this section, we compare the energy performance of the proposed *A-GTA-S* against the corresponding ones of the five benchmark strategies of Section X-A. The pursued threefold goal is to acquire some insight about: (i) the energy reduction

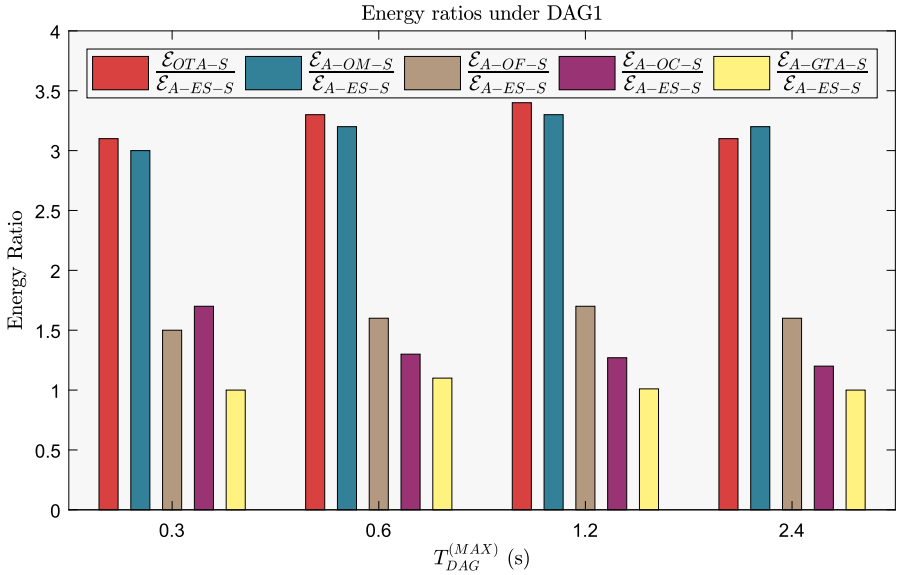

FIGURE 12. Bar plots of the energy ratios under DAG1. All the reported ratios are normalized with respect to the corresponding total energy consumed by *A-ES-S*.

stemming from the *dynamic* optimization of the computing-networking resources versus the corresponding case of *static* resource usage; (ii) the performance gap between the proposed *A-GTA-S* and the exhaustive search-based *A-ES-S*; and, (iii) the energy-saving capability offered by the Mobile-Fog-Cloud *three-tier* computing platform of Fig. 4 versus the only Mobile, Mobile-Cloud and Mobile-Fog corresponding ones.

The obtained numerical results are summarized by the bar plots of Figs. 12, 13, and 14 under *DAG1*, *DAG2*, and *DAG3*, respectively. Their examination gives arise to the following three main sets of remarks.

1) A-GTA-S VERSUS O-TA-S

A number of seminal (even quite recent) contributions [21], [26], [33], [49] tackles with the problem of the resource augmentation of mobile devices by developing various heuristic/meta-heuristic/optimal solutions for energy-efficient task offloading. However, they *neglect* to consider, indeed, the companion problem of the dynamic scaling of the computing and/or network resources. Hence, a key (still open) question concerns how much energy may be actually saved by *jointly* performing task and dynamic resource allocation. By design, a direct comparison of the energy consumed by the *A-GTA-S* and *O-TA-S* provides the response to this question. In this regard, a comparative examination of the red and yellow-colored bars of Figs. 12, 13 and 14 leads to three main insights. First, the energy ratio $\mathcal{E}_{O-TA-S}/\mathcal{E}_{A-GTA-S}$ ranges over the intervals: 3.0 – 3.1, 2.2 – 3.4, and: 3.8 – 3.9 under *DAG1*, *DAG2* and *DAG3*, respectively. Second, under a fixed DAG, the energy savings stemming from performing dynamic resource allocation reach their maxima at values of $T_{DAG}^{(MAX)}$ of the order of 0.6 – 1.2 (s), while tend to somewhat decrease at smaller and higher execution delays. Third, the average energy saving stemming from dynamic optimization is somewhat more relevant under *DAG3*.

Overall, the key lesson stemming from these considerations is that the dynamic optimization of the allocated computing-networking resources plays, indeed, a *major* role in reducing the energy consumption of the simulated platform of Fig. 4.

2) A-GTA-S VERSUS A-ES-S

We pass now to focus on the trade-offs among the energy performance and computational complexity that are attained by the proposed (meta-heuristic) *A-GTA* and the benchmark (optimal) *A-ES* strategies. In this regard, we recall that, in the carried out simulations, the computing complexity of the benchmark *A-ES-S* is about 11 times larger than the corresponding one of the proposed *A-GTA-S* (see the last part of Section X-C). At the same time, a direct inspection of the yellow-colored bars of Figs. 12, 13 and 14 unveils that the energy gaps between the proposed *A-GTA-S* and the benchmark *A-ES-S* maintain below 2% over the full spectrum of the considered maximum DAG execution times. These considerations lead to the conclusion that the tested implementation of the proposed *A-GTA-S* retains, indeed, good performance-vs.-complexity trade-offs against the benchmark *A-ES-S* one.

3) THREE-TIER VERSUS SINGLE/TWO-TIER EXECUTION PLATFORMS

A potential drawback of multi-tier distributed computing platforms is that the number of involved network connections tend to grow with the number of inter-connected tiers, and this could increase the network component of the overall consumed energy. In this regard, we recall that the

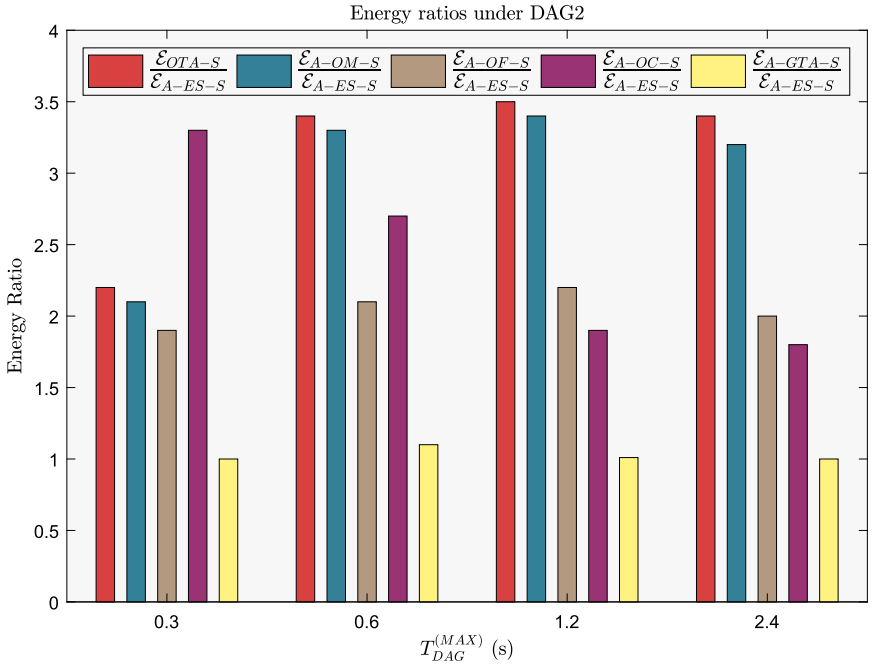

FIGURE 13. Bar plots of the energy ratios under DAG2. All the reported ratios are normalized with respect to the corresponding total energy consumed by *A-ES-S*.

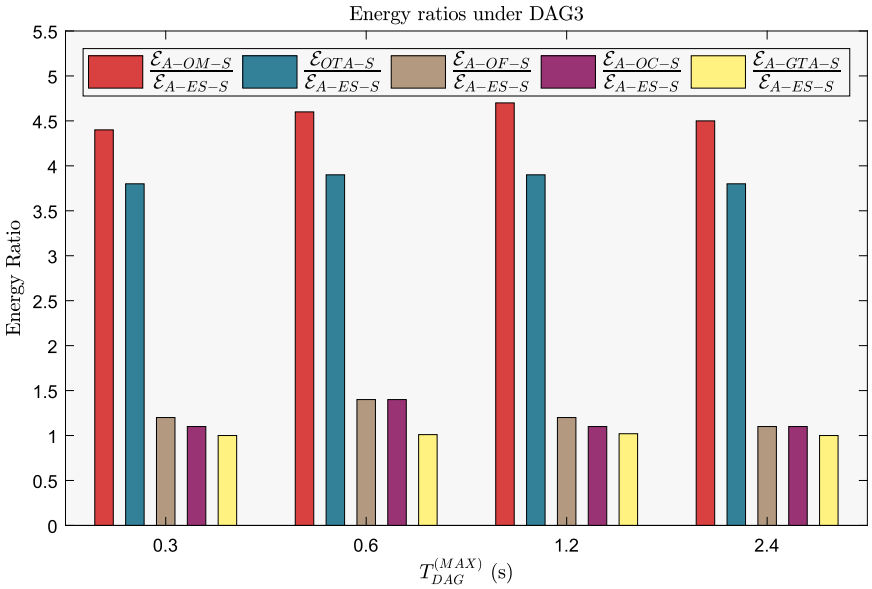

FIGURE 14. Bar plots of the energy ratios under DAG3. All the reported ratios are normalized with respect to the corresponding total energy consumed by *A-ES-S*.

(previously defined) *A-OM*, *A-OF* and *A-OC* benchmark strategies utilize, by design, only the Mobile device and the two-tier Fog-Mobile and Cloud-Mobile platforms for the execution of the application DAGs. Furthermore, all these benchmark strategies perform dynamic scaling of the utilized computing frequencies and wireless network throughput (see their definitions of Section X-A). Hence, in order to attain insight about the net trade-off among the reduction of the computing energy arising from the utilization of multi-tier computing platforms and the corresponding increment of the network energy needed for their inter-connection, it suffices to compare the blue, cyan, magenta, and yellow-colored bars of Figs. 12, 13 and 14. Their comparison leads to two main conclusions.

First, the energy ratio $\mathcal{E}_{A-OM-S}/\mathcal{E}_{A-GTA-S}$ takes values over the intervals: 3.0 – 3.3, 2.1 – 3.4, and 4.4 – 4.7, under *DAG1*, *DAG2* and *DAG3*, respectively. The corresponding intervals of the energy ratios $\mathcal{E}_{A-OF-S}/\mathcal{E}_{A-GTA-S}$, and

$\mathcal{E}_{A-OC-S}/\mathcal{E}_{A-GTA-S}$ are: 1.5 – 1.7, 1.9 – 2.1, 1.1 – 1.4, and: 1.2 – 1.7, 1.8 – 3.3, 1.2 – 1.4, respectively. Hence, in the carried out simulations, the minimum (i.e., worst case) energy-savings guaranteed by the three-tier Mobile-Fog-Cloud platform of Fig. 4 over the Mobile, Mobile-Fog and Mobile-Cloud ones are of the order of: 110%, 10% and 20%, while the corresponding maximum values are around: 370%, 110% and 230%.

Second, at fixed $T_{DAG}^{(MAX)}$, the average energy savings offered by the Mobile-Fog-Cloud platform over the considered benchmark ones tend to be somewhat more substantial under *DAG3*. Intuitively, this is due to the fact that *DAG3* is the hybrid combination of the basic mesh and tree topologies, so that its energy-saving executions tend to take more advantage from the *simultaneous* utilization of all the available Mobile, Fog, and Cloud computing nodes of Fig. 4.

F. SENSITIVITY OF THE A-GTA-S ENERGY PERFORMANCE ON THE DAG COMPUTING-TO-COMMUNICATION RATIOS

The goal of this section is to test the sensitivity of the average energy performance of the proposed *A-GTA-S* on the Computing-to-Communication Ratio (CCR) of the benchmark DAGs of Fig. 5. Formally speaking, the CCR of an application DAG is defined as the ratio between the corresponding per-task average workload and per-edge average weight [1]. Hence, in order to carry out fair energy comparisons, all the tests of this section have been performed by taking the summation of the task workloads and edge weights of each DAG fixed at 4.98 (Mbit), regardless of the actual value assumed by the corresponding CCR.

TABLE 7. Total and network energy consumption (Joule) of the proposed *A-GTA-S* at $CCR = 4, 2, 1$ and 0.5. Case of $\theta_M = \theta_F = \theta_C = 1$ at $T_{DAG}^{(MAX)} = 1.0$ (s). Each reported energy value is averaged over 20 independent runs of Algorithm 2.

CCR	DAG1		DAG2		DAG3	
	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$	$\mathcal{E}_{TOT}$	$\mathcal{E}_{NET}$
4.0	22.54	6.47	22.91	8.10	16.00	5.96
2.0	26.94	10.48	27.31	12.64	19.05	6.55
1.0	27.31	11.12	27.94	13.10	19.51	6.93
0.5	19.64	5.11	21.04	7.55	14.38	3.63

The average total and network energy consumption obtained by running the proposed *A-GTA-S* of Algorithm 2 are reported in Table 7 for values of CCR ranging from 4 (case of computing-intensive DAGs) to 0.5 (case of communication-intensive DAGs). All these results refer to the Eco-centric service model of (38) at $T_{DAG}^{(MAX)} = 1.0$ (s).

An examination of these results unveils that, in all carried out tests, both the total energy $\mathcal{E}_{TOT}$ and the network energy $\mathcal{E}_{NET}$ consumed by the *A-GTA-S* attain their maxima at $CCR = 1$ (i.e., in the case of balanced computing and communication loads), while they decrease at lower and higher CCR values. We have numerically ascertained that this (seemingly unexpected) behavior is, indeed, induced by the considered Eco-centric service scenario. In fact, under this service model, the *JOP* objective function in (37a) accounts for the computing energy of all Mobile-Fog-Cloud nodes of the simulated system of Fig. 4 (see the expression of $\mathcal{E}_{TOT}$ in (32) at $\theta_M = \theta_F = \theta_C = 1$). This triggers the allocation policy followed by the *A-GTA-S* to scatter the DAG workload over *all* the available computing nodes when the CCR values are high, so to reduce the resulting total computing energy. However, decreasing values of CCR increase the consumed network energy, so that, as it could be expected, the increment of the network energy balances the corresponding reduction of the computing one under balanced operating conditions (i.e., at $CCR = 1$). Further decrements of the CCR values induce the allocation policy applied by the *A-GTA-S* to reduce the number of utilized computing nodes, in order to save network energy and, then, lower the resulting total energy.

Overall, two conclusions stem from the carried out analysis. First, under the Eco-centric service model, the most energy demanding operating conditions take place for CCR around the unit, while less energy is wasted at higher and lower values of CCR. Second, a comparison of the first and last rows of Table 7 also points out that computing-intensive operating conditions consume more energy than communication intensive ones under all considered DAGs.

However, we anticipate that these conclusions, could be, indeed, no longer true when the Mobile-centric service model of the next Section X-G is considered.

G. PERFORMANCE SENSITIVITY ON THE ADOPTED SERVICE MODEL

The conclusions of the last section trigger us to further investigate the performance sensitivity of the proposed *A-GTA-S* on the actually adopted service model.

For this purpose, we have considered the test *DAG4* reported in Fig. 15, that is typically considered in the literature for comparing task allocation policies under different service models [33]. Specifically, *DAG4* details the workflow of a real-world video navigation program for mobile stream application. It involves the parallel execution of three sub-programs, namely a graphic sub-program (left section of Fig. 15), a subprogram for face detection (middle section of Fig. 15), and a video-processing subprogram (right section of Fig. 15). All these sub-programs share the same input and output nodes (i.e., nodes 1 and 15 in Fig. 15), that implement data-rendering functionalities and, then, are executed by the Mobile device. *DAG4* is a quite large-size DAG composed of 15 tasks and 21 edges. Its topology retains the following features that make it a challenging test DAG: (i) it is asymmetric; (ii) it is composed of the parallel combination of three heterogeneous sub-DAGs, that exhibit fork, parallel and tree-shaped topologies; and, (iii) the face detection and video processing subprograms are computing and communication-intensive, while the graphic subprogram is of mixed type.

Tables 8 and 9 report the simulated performance of the proposed *A-GTA-S* under the Eco-centric and Mobile-centric service models of (38) and (39), respectively. All the reported

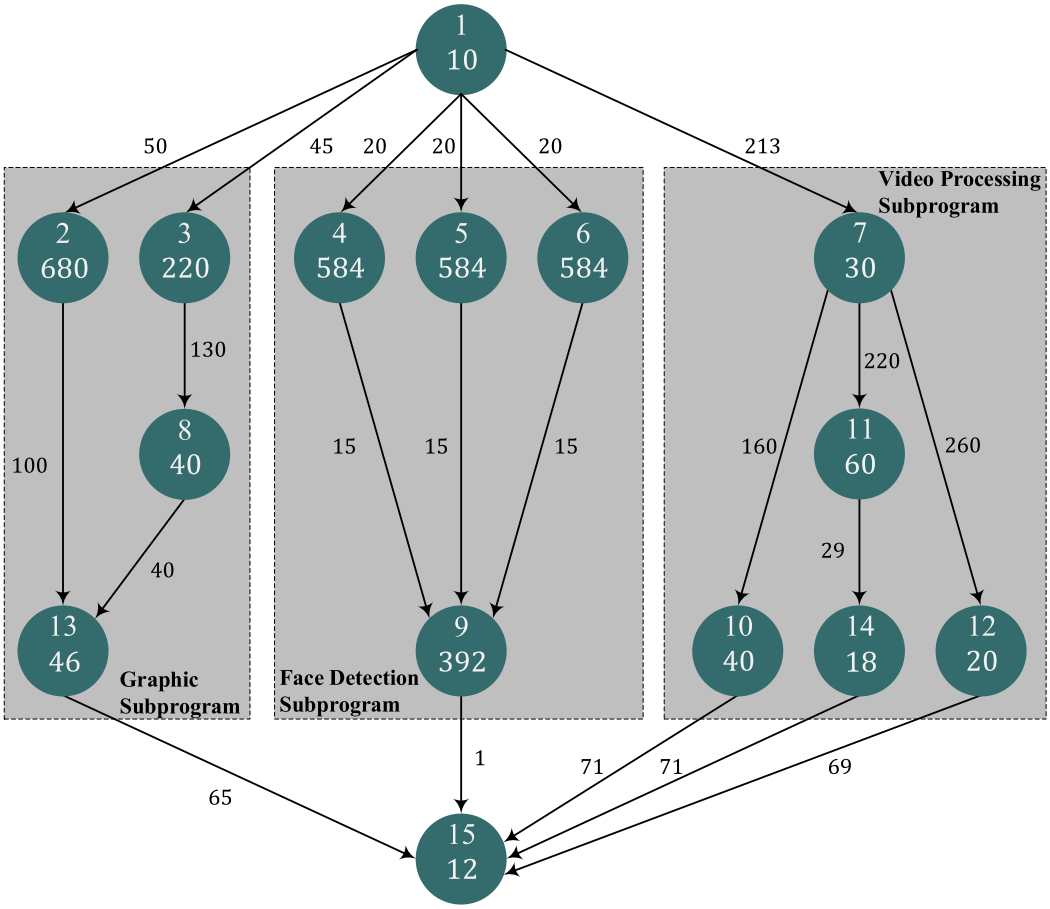

FIGURE 15. Test *DAG4* describing a real-world stream application of radio-navigation. Task workloads and edge weights are in (kbit).

numerical results have been obtained by running Algorithm 2 at $PS = 120$ and $G_{MAX} = 20$, and each one refers to the best (i.e., minimum-energy) outcome obtained over 10 independent runs. An examination of these results leads to three main insights.

First, a comparison of the task allocation patterns reported in the second columns of Tables 8 and 9 unveils that, in average, the workload allocated to the Mobile device under the Eco-centric service model is about 2.5 times larger than the corresponding one under the Mobile-centric case. This is a (first) direct consequence of the fact that both the computing and network resources are made available for-free under the Mobile-centric framework.

Second, a comparison of the numerical values reported in the last columns of Tables 8 and 9 points out that the total energy $\mathcal{E}_{A-GTA-TOT}$ consumed by the overall platform of Fig. 4 under the Mobile-centric framework is about 58.0%, 45.0%, 38.0% and 26.0% larger than the corresponding one of the Eco-centric case at $T_{DAG}^{(MAX)} = 0.3, 0.6, 1.2$, and 2.4 (s), respectively.

Third, the computing-plus-networking energy $\mathcal{E}_{A-GTA-MOB}$ consumed by the Mobile device under the Eco-centric framework is about 40.0%, 31.0%, 22.0% and 17.0% larger than the corresponding one under the Mobile-centric case. In this regard, we have also numerically ascertained that, in the Mobile-centric case, the network component of the overall profiled energy $\mathcal{E}_{A-GTA-MOB}$ is substantial, and of the order of about: 86.8%, 85.0%, 72.2% and 61.0% at $T_{DAG}^{(MAX)} = 0.3, 0.6, 1.2$, and 2.4 (s), respectively.

Overall, the ultimate lesson, which stems from the above discussion, is that both the energy consumption and the task allocation patterns strongly depend on the actually adopted service model.

H. AVERAGE ENERGY PERFORMANCE OF MULTI/SINGLE-TIER ECOSYSTEMS UNDER RANDOMLY TIME-VARYING WiFi CONNECTIVITY

The goal of this section is twofold. First, we aim at indagating on the sensitivity of the average energy performance of the simulated ecosystem of Fig. 4 when, due to device mobility and limited coverage of the Fog node, the availability of the Mobile-Fog WiFi connection alternates ON-OFF periods in a random way. A related consideration is that the number of involved communication links tend to grow with the number of inter-connected tiers, and this may lead, in turn, to an increment of the network component of the overall

TABLE 8. Task allocations and energy consumption of the proposed A-GTA strategy under DAG4. Eco-centric case of $\theta_M = \theta_F = \theta_C = 1$. $T_{DAG}^{(MAX)}$ is measured in (s) while energy is measured in (Joule).

$T_{DAG}^{(MAX)}$	Mobile Tasks IDs	Fog Tasks IDs	Cloud Tasks IDs	$\mathcal{E}_{A-GTA-MOB}$	$\mathcal{E}_{A-GTA-TOT}$
0.3	$\{1, 7, 10-12, 14, 15\}$	$\{2-6, 8, 9, 13\}$	$\{-\}$	10.33	27.10
0.6	$\{1, 7, 10-12, 14, 15\}$	$\{2, 3, 8, 13\}$	$\{4-6, 9\}$	9.24	23.63
1.2	$\{1, 7, 10-12, 14, 15\}$	$\{-\}$	$\{2-6, 8, 9, 13\}$	8.12	21.73
2.4	$\{1, 7, 9-15\}$	$\{-\}$	$\{2-6, 8\}$	7.41	21.10

TABLE 9. Task allocations and energy consumption of the proposed A-GTA strategy under DAG4. Mobile-centric case of $\theta_M = 1$ and $\theta_F = \theta_C = 0$. $T_{DAG}^{(MAX)}$ is measured in (s) while energy is measured in (Joule).

$T_{DAG}^{(MAX)}$	Mobile Tasks IDs	Fog Tasks IDs	Cloud Tasks IDs	$\mathcal{E}_{A-GTA-MOB}$	$\mathcal{E}_{A-GTA-TOT}$
0.3	$\{1, 14, 15\}$	$\{2-8, 10-13\}$	$\{9\}$	7.38	42.66
0.6	$\{1, 14, 15\}$	$\{3-8, 10-13\}$	$\{2, 9\}$	7.05	38.61
1.2	$\{1, 14, 15\}$	$\{3, 5-8, 10-13\}$	$\{2, 4, 9\}$	6.65	30.01
2.4	$\{1, 12, 14, 15\}$	$\{-\}$	$\{2-11, 13\}$	6.33	26.63

TABLE 10. Average energy consumption (Joule) of the proposed *A-GTA-S* strategy under randomly time-variant availability of the up/down WiFi connections. Case of *DAG4* at $\theta_M = \theta_F = \theta_C = 1$ and $T_{DAG}^{(MAX)} = 1.0$ (s).

AV_{WiFi}	Average $\mathcal{E}_{A-GTA-MOB}$	Average $\mathcal{E}_{A-GTA-TOT}$	Average $\mathcal{E}_{A-GTA-NET}$
1.00	7.83	22.17	7.51
0.75	8.40	23.70	6.10
0.50	9.01	25.10	4.82
0.25	10.91	26.50	3.41
0.00	14.33	28.98	2.51

consumed energy. Therefore, a second goal of this section is to investigate about the net trade-off among the reduction of the computing energy arising from the utilization of multi-tier computing nodes and the corresponding increment of the network energy needed for their inter-connection.

In order to meet this twofold goal, we have numerically evaluated the energy performance of the ecosystem of Fig. 4 under *DAG4* of Fig. 15 at $CCR = 2, \theta_M = \theta_F = \theta_C = 1$, and $T_{DAG}^{(MAX)} = 1.0$ (s).

Specifically, in the simulated scenario considered here:

i. the up/down Mobile-Cloud cellular connection is permanently ON;
ii. the up/down Mobile-Fog WiFi connection is available only during a fraction $AV_{WiFi} \in [0, 1]$ of the DAG execution time; and,
iii. the corresponding up/down WiFi throughput $R_{M\to F}$ and $R_{F\to C}$ are modeled as two unit-correlated random variables, whose probability density functions (PDFs) are uniform over the corresponding allowed intervals: $\left[0, R_{M\to F}^{MAX}\right]$ and $\left[0, R_{F\to M}^{MAX}\right]$, and present two Dirac's spikes of areas: $(1 - AV_{WiFi})$ at the origin.

The second column of Table 10 reports the energy consumption of the Mobile device, while the third and fourth columns give the total energy wasted by the overall simulated ecosystem of Fig. 4, together with corresponding network energy. In order to account for the random nature of the simulated WiFi connections, each value of Table 10 is the average over 50 independent runs of the proposed *A-GTA-S*.

1) EFFECTS OF THE INTERMITTENT WIFI AVAILABILITY

A comparative examination of the columns Table 10 points out that the effects of the availability AV_{WiFi} of the WiFi connection on the reported energy are substantially different. Specifically, by passing from $AV_{WiFi} = 1$ (i.e., both Fog

and Cloud nodes are permanently available for DAG execution) to $AV_{WiFi} = 0$ (i.e., only the Cloud node is available for DAG execution), we experience that: (i) the average computing-plus-network energy $\mathcal{E}_{A-GTA-MOB}$ consumed by the Mobile device increases of about 83.0% (see the second column of Table 10); (ii) the average total computing-plus-communication energy $\mathcal{E}_{A-GTA-TOT}$ of the overall ecosystem of Fig. 4 increases by about 30.7% (see the third column of Table 10); and, (iii) the corresponding average network energy $\mathcal{E}_{A-GTA-NET}$ consumed by the overall ecosystem decreases by about 199.0% (see the last column of Table 10). The common rationale behind these trends is that the execution of more and more tasks are shifted from the Fog node to the Mobile and Cloud ones for decreasing values of AV_{WiFi}. As a matter of this trend, the volume of the total inter-node traffic decreases, but the computing components of the energy wasted by both the Mobile device and the overall ecosystem increase.

2) SINGLE-TIER VERSUS MULTI-TIER ECOSYSTEMS

In order to further corroborate this trend, we have also tested that the corresponding average energy consumption of the (previously introduced) *A-OM* strategy is $\mathcal{E}_{A-OM} = 89.91$ (Joule) under the same simulated setting. In this regard, we point out that:

i. since, by design, the *A-OM* strategy utilizes only the Mobile device for task execution, this strategy features the performance of a *single-tier* execution platform, whose total energy consumption $\mathcal{E}_{A-OM}$ equates to the corresponding computing energy (i.e., by design, $\mathcal{E}_{A-OM-NET}$ is vanishing);
ii. since the proposed *A-GTA* strategy exploits, by design, all the actually available computing nodes for task placement, the returned energy $\mathcal{E}_{A-GTA-TOT}$ of Table 10 at $AV_{WiFi} = 0$ (resp., $AV_{WiFi} = 1$) captures the energy consumption of the two-tier Mobile-Cloud (resp., three-tier Mobile-Fog-Cloud) execution platform embedded in Fig. 4.

Overall, on the basis of these remarks, we conclude that the average computing-plus-networking total energy consumption of the simulated ecosystem of Fig. 4 increases for decreasing number of the exploited tiers, and equates to, indeed, 22.17, 28.98, and 89.91 (Juole) when three tiers, two tiers and a single tier are activated, respectively.

This final conclusion provides further full-fledged support both for the multi-tier networked design approach and the Eco-centric perspective pursued by our work.

XI. CONCLUSION AND HINTS FOR FUTURE RESEARCH

It is expected that the convergence of Fog Computing, Cloud Computing and multi-radio 5G technology allows resource-limited smartphones to support throughput-sensitive mobile stream applications in an energy efficient way. Motivated by this expectation, in this paper, we develop and discuss the main implementation aspects of *EcoMobiFog*, a technological platform for the adaptive joint optimization of the resource allocation and task offloading in 5G-networked virtualized ecosystems composed by an arbitrary number of Fog/Cloud nodes. The energy-delay performance of the solving framework implemented by *EcoMobiFog* is numerically evaluated and compared with respect to the corresponding ones of some state-of-the-art benchmark solutions under a number of operative scenarios that embrace both Eco and Mobile-centric service models.

Being the overall afforded topic still in its infancy, we believe that the presented results could be extended along (at least) four main research directions.

First, the developed solving approach reflects the basic features of the current Middleware management platforms, in which the task scheduling discipline is statically assigned at the compiling time. Hence, including in the afforded *JOP* formulation also the dynamic optimization of the task-execution ordering followed by the computing nodes may be a first research direction of potential interest. The main expected challenge stems from the fact that the dynamic optimization of the task scheduling discipline has been recently proved to be an NP-hard integer-valued problem, even in the basic case of fixed resource allocation [33].

A second hint for future research arises from the consideration that 5G technology adopts, by design, massive numbers of transmit/receive antennas at the terminals [50]. Hence, including the effects of space-time coding and spatial multiplexing [51] in the energy models of Section V-B may be valuable.

A third future research line moves from the consideration that the adaptive framework of Sections VII-B and VII-C for the dynamic adjustments of the utilized networking-plus-computing resources is purely reactive, i.e., it does not exploit any form of forecasting of the mobility-affected environmental conditions. Including in the solving framework proactive optimization tools that are capable of predicting future resource utilization [52] could be a further research line of potential interest.

Finally, it could be worthwhile to carry out the implementation of a (small-scale) test-bed of the overall proposed *EcoMobiFog* technological platform of section IX, in order to check its performance through real-world field-trials. This is, indeed, the ultimate goal of the (ongoing) *GAUChO* research project (see https://www.gaucho.unifi.it), that provides the reference framework of this work.

APPENDIX A
MAIN TAXONOMY AND SIMULATED SETUP

The Table 11 in the sequel reports the main symbols used in this paper, their meaning/role, measuring units and simulated values.

APPENDIX B
PROOF OF THE CONDITION FOR THE JOP FEASIBILITY

The proof of Proposition 1 exploits some basic formal properties of T_{DAG} that are reported in the following Lemma 1:

Lemma 1 (Formal Properties of T_{DAG}): Let the assumptions on T_{DAG} of Section IV-C be met. Then, we have that:

a. T_{DAG} is a jointly convex function of the $(3Q+4)$ scalar optimization variables gathered by the resource vector $\overrightarrow{RS}$ in (36);
b. T_{DAG} is a non-decreasing function of the task sizes: $\{s_i, i=1,\ldots,V\}$ and edge weights $\left\{d_{ij}\,(i,j)\in E\right\}$ of the considered application DAG. Furthermore, T_{DAG} is a non-increasing function of the processing capacities: $\{n_N f_N, N\in\mathcal{A}\}$ of the computing nodes and the transport throughput: $\left\{R_{N_1\rightarrow N_2}, N_1\neq N_2; N_1, N_2\in\mathcal{A}\right\}$ of the underlying network connections. ■

Proof:

a) According to the assumptions reported in Sections IV-A and IV-B, each per-task service time $T_{N,i}^{(SER)}$ (resp., per-task network delay $T_{N,i}^{(NET)}$) is convex with respect to the corresponding computing frequency f_N (resp., the connection throughput $R_{N_1\rightarrow N}$), because, by design, it scales as $1/f_N$ (resp., $1/R_{N_1\rightarrow N}$). Hence, each per-task execution time $T_{N,i}^{(EXE)}$ in (13) is also convex in the involved optimization variables, because it is the summation of two convex functions. As a consequence, since T_{DAG} is, by assumption, a jointly convex and non-decreasing composition of convex functions (see Section IV-C), it is jointly convex in the optimization variables gathered by the resource vector $\overrightarrow{RS}$ in (36).
b) By design, each per-task service time $T_{N,i}^{(SER)}$ (resp., each per-task network delay $T_{N,i}^{(NET)}$) does not decrease for increasing task sizes $\{s_i, i=1,\ldots,V\}$ (resp., edge weights $\left\{d_{ij}\,(i,j)\in E\right\}$), while it does not increase for increasing processing capacities $\{n_N f_N, N\in\mathcal{A}\}$ (resp., transport throughput $\left\{R_{N_1\rightarrow N_2}, N_1\neq N_2; N_1, N_2\in\mathcal{A}\right\}$) (see the assumptions of Sections IV-A and IV-B). Hence, being the summation of the corresponding per-task service time and network delay, the same monotonic properties are also retained by the resulting per-task execution time $T_{N,i}^{(EXE)}$ in (13). As a consequence, by assumption, T_{DAG} is not decreasing with respect to each per-task execution time (see Section IV-C), the validity of the stated monotonic properties directly follows. □

By leveraging the stated formal properties of T_{DAG}, we note that:

i. $Tmax^{(SER)}$ in (44) is a feasible upper bound on all task execution times. This is due to the fact that $Tmax^{(SER)}$ is computed by jointly considering the maximum task size (see the numerator of (31)), together with the minimum per-task fraction of the per-node computing frequency and the minimum of the allowed per-node maximum processing frequencies (see the product at the denominator of (44)); and,
ii. $Tmax^{(NET)}$ in (45) is a feasible upper bound on all network times. This is due to the fact that $Tmax^{(NET)}$ is computed by jointly considering the maximum volume of the per-task input data and the maximum network failure factor (see the product at the numerator of (45)), together with the minimum of the per-connection maximum throughput (see the denominator of (45)).

As a consequence, the resulting $T_{EXE}^{(MAX)}$ in (46) constitutes, by design, a feasible upper bound on the set of the per-task execution times. Hence, the validity of the feasibility condition in (47) directly arises from the not decreasing behavior of T_{DAG} with respect to the per-task execution times (see Section IV-C).

Before proceeding, two explicative remarks about the meaning/role of the reported feasibility condition are in order.

First, the sufficiency of this condition stems from the fact that it considers the *worst* case in which the task of maximum size is also the task whose execution requires the maximum volume of input data (see $s^{(MAX)}$ and $w_{IN}^{(MAX)}$ at the numerators of (44) and (45), respectively). Second, the evaluation of $T_{DAG}^{(UP)}$ in (47) may be carried out in closed-form by exploiting only the defining parameters of the considered *JOP*. Just as application examples, in the case of the (previously introduced) Sequential service and scheduling disciplines, $T_{DAG}^{(UP)}$ is computed as in (78), as shown at the bottom of this page, while, in the case of intra-node WPS service discipline and inter-node Parallel Task Scheduling discipline, (79), as shown at the bottom of this page, holds.

APPENDIX C
PROOF OF THE RAP CONVEXITY

The proof of the *RAP* convexity relies on the formal properties of the per-node computing energy and per-connection wireless network energy proved in the following Lemma 2 and Lemma 3, respectively.

$$T_{DAG}^{(UP)} \equiv \sum_{i=1}^{V} T_{EXE}^{(MAX)} = V\left(\frac{s^{(MAX)}}{\min\limits_{N\in\mathcal{A}}\left\{n_N f_N^{(MAX)}\right\}} + \frac{w_{IN}^{(MAX)}\left(1+\max\limits_{N_1,N_2\in\mathcal{A}}\left\{\overline{NF}_{N_1\rightarrow N_2}\right\}\right)}{\min\limits_{N_1,N_2\in\mathcal{A}}\left\{R_{N_1\rightarrow N_2}^{(MAX)}\right\}}\right), \quad SEQ-STS \tag{78}$$

$$\begin{aligned} T_{DAG}^{(UP)} &\equiv \max_{1\leq i\leq V}\left\{T_{EXE}^{(MAX)}\right\} \\ &= \left(\frac{s^{(MAX)}}{\left(\frac{\min\limits_{1\leq i\leq V}\{\phi_i\}}{\sum_{j=1}^{V}\phi_j}\right)\times\min\limits_{N\in\mathcal{A}}\left\{n_N f_N^{(MAX)}\right\}} + \frac{w_{IN}^{(MAX)}\left(1+\max\limits_{N_1,N_2\in\mathcal{A}}\left\{\overline{NF}_{N_1\rightarrow N_2}\right\}\right)}{\min\limits_{N_1,N_2\in\mathcal{A}}\left\{R_{N_1\rightarrow N_2}^{(MAX)}\right\}}\right), \quad WPS-PTS \end{aligned} \tag{79}$$

TABLE 11. List of the main parameters, their meaning/role, measuring units and simulated values.

Parameter	Meaning/Role	Measuring Units	Simulated Settings
Q	Number of Fog nodes	Dimensionless	$Q = 1$
$\mathcal{A} \stackrel{\text{def}}{=} \{M, F_1, \ldots, F_Q, C\}$	Set of the available computing nodes	Dimensionless	$\mathcal{A} = \{M, F, C\}$
$BHS \stackrel{\text{def}}{=} \{F_1, \ldots, F_Q, C\}$	Set of nodes of the Backhaul network	Dimensionless	$BHS = \{F, C\}$
$n_N, N \in \mathcal{A}$	Number of the (virtual) computing cores equipping the computing node N	Dimensionless	$n_M = 1, n_F = 4, n_C = 12$
$\overline{NF}_{N' \to N''}$	Average number of failures of the connection from computing node N' to computing node N''	Dimensionless	$\overline{NF}_{M \to F} = \overline{NF}_{F \to M} = 1.1$ $\overline{NF}_{M \to C} = \overline{NF}_{C \to M} = 0.1$ $\overline{NF}_{F \to C} = \overline{NF}_{C \to F} = 0.01$
V	Number of tasks of the application DAG	Dimensionless	$V \geq 9$
$\vec{x} = [x_1, \ldots, x_V]$	Vector of task allocation, with component $x_i \in \mathcal{A}$	Dimensionless	Optimization variable
$f_N, N \in \mathcal{A}$	Per-core computing frequency at the computing node N	bit/s	Optimization variable
$f_N^{(MAX)}, N \in \mathcal{A}$	Per-core maximum computing frequency at the computing node N	bit/s	$f_M^{(MAX)} = 12 \times 10^6$ $f_F^{(MAX)} = 12 \times 10^6$ $f_C^{(MAX)} = 12 \times 10^6$
$\mathcal{E}_N, N \in \mathcal{A}$	Computing energy consumed by the device clone at node N	Joule	To be optimized
$R_{M \to N}$	Up throughput of the TCP/IP connection from the Mobile to the Cloud/Fog node $N \in BHS$	bit/s	Optimization variable
$R_{N \to M}$	Down throughput of the TCP/IP connection from the Cloud/Fog node $N \in BHS$ to the Mobile	bit/s	Optimization variable
$\mathcal{E}_{M \to N}, \mathcal{E}_{N \to M}$	Energy consumed by the one-way wireless connections: $M \to N$, and $N \to M$, with $N \in BHS$	Joule	To be optimized
$R_{M \to N}^{(MAX)}$	Maximum throughput of the TCP/IP connection from the Mobile to the Cloud/Fog node $N \in BHS$	bit/s	$R_{M \to F}^{(MAX)} = 8.0 \times 10^6$ $R_{M \to C}^{(MAX)} = 6.5 \times 10^6$
$R_{N \to M}^{(MAX)}$	Maximum throughput of the TCP/IP connections from the Cloud/Fog node $N \in BHS$ to the Mobile	bit/s	$R_{F \to M}^{(MAX)} = 9.0 \times 10^6$ $R_{C \to M}^{(MAX)} = 7.0 \times 10^6$
$R_{N_1 \leftrightarrow N_2}$	Throughput of the backhaul TCP/IP two-way connection between nodes N_1, N_2, with $N_1 \neq N_2$, and $N_1, N_2 \in BHS$	bit/s	$R_{C \leftrightarrow F} = 3.7 \times 10^6$
$\mathcal{E}_{N_1 \leftrightarrow N_2}$	Energy consumed by the two-way backhaul connection: $N_1 \leftrightarrow N_2$, with $N_1 \neq N_2$, and $N_1, N_2 \in BHS$	Joule	To be optimized
T_{DAG}	Total execution time of the considered application DAG	s	To be optimized
$T_{DAG}^{(MAX)} \stackrel{\text{def}}{=} \frac{1}{TH_0^{(MIN)}}$	Per-DAG maximum allowed execution time	s	$0.3 \leq T_{DAG}^{(MAX)} \leq 2.4$
$T_N^{(SER)}, (N \in \mathcal{A})$	Service time of node N	s	To be optimized
$T_{i,N}^{(EXE)}$, $N \in \mathcal{A}, i = 1, \ldots, V$	Execution time of the i-th task at node N	s	To be optimized
$\theta_N \in \{0, 1\}$	Binary parameter. It is zero (resp., unit valued) if the computing-plus-networking energy consumed by the computing node $N \in \{M, F, C\}$ is (resp., is not) for free	Dimensionless	$\theta_M = \theta_F = \theta_C = 1$ $\theta_M = 1, \theta_F = \theta_C = 0$

TABLE 11. ***(Continued.)*** **List of the main parameters, their meaning/role, measuring units and simulated values.**

nc_N	Number of containers simultaneously running atop the CPU at the computing node $N \in \mathcal{A}$	Dimensionless	$nc_M = 1$ $nc_F = 16$ $nc_C = 24$
γ_N	Positive exponent of the dynamic power consumption of the CPU at the computing node $N \in \mathcal{A}$	Dimensionless	$\gamma_M = 3.2$ $\gamma_F = 3.1$ $\gamma_C = 3.0$
k_N	Positive scaling factor profiling the dynamic power consumption of the CPU at the computing node $N \in \mathcal{A}$	$\frac{\text{Watt}}{(\text{bit/s})^{\gamma_N}}$	$k_M = 7.50 \times 10^{-21}$ $k_F = 9.78 \times 10^{-20}$ $k_C = 1.14 \times 10^{-19}$
r_N	Fraction of the overall computing power shared by the cores at the computing node $N \in \mathcal{A}$	Dimensionless	$r_M = 0.0$ $k_F = 0.2$ $k_C = 0.1$
$\mathcal{P}^{(IDLE)}_{CPU-N}$	Power consumed in the idle state by the physical CPU at the computing node $N \in \mathcal{A}$	Watt	$\mathcal{P}^{(IDLE)}_{CPU-M} = 1.2$ $\mathcal{P}^{(IDLE)}_{CPU-F} = 220$ $\mathcal{P}^{(IDLE)}_{CPU-C} = 440$
$\mathcal{P}^{(IDLE)}_{BHNET-N}$	Power consumed in the idle state by each physical Ethernet NIC at the Fog and Cloud nodes	Watt	$\mathcal{P}^{(IDLE)}_{BHNET-F} = \mathcal{P}^{(IDLE)}_{BHNET-C} = 10^{-14}$
$\mathcal{P}^{(IDLE)}_{SRNET-N-k}$	Power consumed in the idle state by the k-th short-range physical NIC at node N, with $N \in \{M, F_1, \ldots, F_Q\}$ and $k = 1, \ldots, Q$	Watt	$\mathcal{P}^{(IDLE)}_{SRNET-M} = \mathcal{P}^{(IDLE)}_{SRNET-F} = 1.3$
$\mathcal{P}^{(IDLE)}_{LRNET}$	Power consumed in the idle state by each physical long-range NIC at the Mobile device and Cloud node	Watt	$\mathcal{P}^{(IDLE)}_{LRNET} = 0.82$
$\xi^{(Tx)}_{(N_1,N_2)}$	Positive exponent of the dynamic power consumption of the wireless NIC connecting the computing nodes N_1 and N_2 and operating in the transmit mode	Dimensionless	$\xi^{(Tx)}_{(M,F)} = 2.40$ $\xi^{(Tx)}_{(M,C)} = 2.45$
$\xi^{(Rx)}_{(N_1,N_2)}$	Positive exponent of the dynamic power consumption of the wireless NIC connecting the computing nodes N_1 and N_2 and operating in the receive mode	Dimensionless	$\xi^{(Rx)}_{(M,F)} = 2.20$ $\xi^{(Rx)}_{(M,C)} = 2.34$
η	Positive exponent of the RTTs of the short and long-range TCP/IP wireless connections	Dimensionless	$\eta = 0.6$
$RTT_{(M,N)}$	Average RTT of the TCP/IP short-range connection between the Mobile device and Fog node $N \in \{F_1, \ldots, F_Q\}$	s	$RTT_{(M,F)} = 1.0 \times 10^{-3}$
$RTT_{(M,C)}$	Average RTT of the Mobile-Cloud TCP/IP long-range cellular connection	s	$RTT_{(M,C)} = 1.0 \times 10^{-2}$
$RTT_{(N_1,N_2)}$	Average RTT of the (possibly, multi-hop) two-way TCP/IP backhaul connection between nodes N_1 and N_2, with $N_1 \neq N_2$ and $N_1, N_2 \in BHS$	s	$RTT_{(N_1,N_2)}$
$MSS^{(N_1,N_2)}$	Maximum size of a TCP segment of the TCP/IP connection between the computing nodes N_1 and N_2, with $N_1 \neq N_2$ and $N_1, N_2 \in \mathcal{A}$	bit	$MSS = 12.0 \times 10^3$
$Pr^{(N_1,N_2)}_{LOSS}$	Loss probability of the TCP/IP backhaul connection between nodes N_1 and N_2, with $N_1 \neq N_2$ and $N_1, N_2 \in BHS$	Dimensionless	$Pr^{(N_1,N_2)}_{LOSS} = 1.56 \times 10^{-5}$
$\Omega^{(Tx)}_{(N_1,N_2)}$	Power profile of the wireless NIC connecting the computing nodes N_1, N_2 in the transmit mode	$\frac{\text{Watt}}{(\text{bit/s})^{\xi^{(Tx)}_{(N_1,N_2)}} \times (\text{s})^{\eta}}$	$\Omega^{(Tx)}_{(M,F)} = 5.00 \times 10^{-14}$ $\Omega^{(Tx)}_{(M,C)} = 2.31 \times 10^{-13}$
$\Omega^{(Rx)}_{(N_1,N_2)}$	Power profile of the wireless NIC connecting the computing nodes N_1, N_2 in the receive mode	$\frac{\text{Watt}}{(\text{bit/s})^{\xi^{(Rx)}_{(N_1,N_2)}} \times (\text{s})^{\eta}}$	$\Omega^{(Rx)}_{(M,F)} = 1.40 \times 10^{-14}$ $\Omega^{(Rx)}_{(M,C)} = 8.10 \times 10^{-15}$

TABLE 11. *(Continued.)* **List of the main parameters, their meaning/role, measuring units and simulated values.**

I_{MAX}	Maximum number of primal-dual iterations performed by the *RAP*	Dimensionless	$500 \leq I_{MAX} \leq 700$
a_{MAX}	Clipping factor of the *RAP* iterations	Dimensionless	$10^{-7} \leq a_{MAX} \leq 8.0 \times 10^{-7}$
$no_{HOP}^{(N_1,N_2)}$	Number of hops of the backhaul connection between nodes N_1 and N_2, with $N_1 \neq N_2$ and $N_1, N_2 \in BHS$	Dimensionless	$no_{HOP}^{(C,F)} = 4$
$\mathcal{P}_{HOP}^{(N_1 \to N_2)}$	One-way per-hop average power consumed by the backhaul connection between nodes N_1 and N_2, with $N_1 \neq N_2$, and $N_1, N_2 \in BHS$	Watt	$\mathcal{P}_{HOP}^{(C \to F)} = 2.85 \times 10^{-1}$
PS	Population size of the Genetic algorithms	Dimensionless	$4 \leq PS \leq 120$
CF	Fraction of the population size that undergoes Genetic crossover	Dimensionless	$CF = 0.5$
G_{MAX}	Number of generations run by the Genetic algorithms	Dimensionless	$10 \leq G_{MAX} \leq 20$
MN	Number of elements of each task allocation vector that undergo Genetic mutation	Dimensionless	$MN = \text{round}\left((V-2)/2\right)$

Lemma 2 (On the Convexity of the Per-Node Computing Energy): Let the assumptions on T_{DAG} of Section IV-C be met. Furthermore, let the exponents of the computing energy of (21) meet the following inequality:

$$\gamma_N \geq 2, \quad N \in \mathcal{A}, \tag{80}$$

Then, each computing energy $\mathcal{E}_N, N \in \mathcal{A}$, is jointly convex in the $(3Q+4)$ scalar optimization variables gathered by the resource vector $\overrightarrow{RS}$ in (36). ■

Proof: Since $\mathcal{E}_N$ is the summation of a static part $\mathcal{E}_N^{(STA)}$ and a dynamic one $\mathcal{E}_N^{(DYN)}$, it suffices to separately prove the convexity of $\mathcal{E}_N^{(STA)}$ and $\mathcal{E}_N^{(DYN)}$. In this regard, we note that:

i. an inspection of the first term on the RHS of (21) points out that $\mathcal{E}_N^{(STA)}$ depends on the involved optimization variables only through the DAG execution time T_{DAG}, whose convexity has been already proved by Lemma 1;
ii. $\mathcal{E}_N^{(DYN)}$ depends on the computing frequency f_N through the product: $(f_N)^{\gamma_N} T_N^{(SER)}$. Hence, since $T_N^{(SER)}$ scales, by assumption, as: $1/f_N$ (see Section IV-B), the above product scales as: $(f_N)^{\gamma_N - 1}$, which, in turn, is a convex function in f_N for $\gamma_N \geq 2$.

This completes the proof. □

Lemma 3 (On the Convexity of the Network Energy of the Wireless Connections): Let the assumptions on T_{DAG} of Section IV-C be met. Furthermore, let the exponents of the wireless network power of (28) meet the following inequalities:

$$\xi_{(M,N)}^{(Tx)} \geq 2, \text{ and } \xi_{(M,N)}^{(Rx)} \geq 2, \quad N \in BHS. \tag{81}$$

Then, the network energy $\mathcal{E}_{M \to N}$ and $\mathcal{E}_{N \to M}$, $N \in BHS$, of each up/down wireless connection are jointly convex in the $(3Q+4)$ scalar optimization variables gathered by the resource vector $\overrightarrow{RS}$ in (36). ■

Proof: After noting that each wireless network energy is still the summation of a static and dynamic component (see Section V-B), the proof can be carried out by replicating the same steps already reported for the proof of Lemma 2. □

About the backhaul network of Fig. 1, we point out that the total energy $\mathcal{E}_{BH-NET}$ in (35) consumed by the (two-way) backhaul connections: $\{N_1 \leftrightarrow N_2, N_1 \neq N_2, N_1, N_2 \in BHS\}$ depends on the involved optimization variables only through T_{DAG} (see the last part of Section V-B), that, in turn, is guaranteed to be convex by Lemma 1.

In order to formally prove the convexity of the *RAP*, it suffices to note that:

i. under any given task allocation vector $\vec{x}$, the objective function: $\mathcal{E}_{TOT}$ in (37a) of the *RAP* is a linear superposition (with positive coefficients) of energy terms, whose convexity are guaranteed by the results of Lemmas 1, 2 and 3; and,
ii. the convexity of T_{DAG} assures that the inequality-type constraint in (37b) on the required application throughput is convex.

The proof of Proposition 3 is now complete.

APPENDIX D
PROOF OF THE CONDITION FOR THE RAP FEASIBILITY

Let us proceed to prove the sufficient and necessary parts of the *RAP* feasibility condition of Proposition 4.

Sufficient part – Let us assume that the condition in (59) is met under the assigned task allocation vector $\vec{x}$. This means, in turn, that $\overrightarrow{RS}^{(MAX)}$ also meets this condition. Since, by design, $\overrightarrow{RS}^{(MAX)}$ also meets all the box constraints in (37c) – (37e) on the maximum allowed resources, it is a feasible solution of the *RAP*, and, then, the *RAP* is feasible.

Necessary part – The proof is by contradiction. Hence, let us assume that the condition in (59) is not met under the assigned task allocation vector $\vec{x}$. Since Lemma 1 guarantees that T_{DAG} is a non-increasing function of each component of the resource allocation vector, we would increase at least

a component of $\overrightarrow{RS}^{(MAX)}$, in order to decrease the value of T_{DAG} and, then, attempt to meet the inequality in (59). However, so doing, at least one of the frequency/throughput resources would violate its upper bound and this would give rise to an infeasible resource allocation vector. This proves, in turn, that the condition in (59) is necessary for the *RAP* feasibility.

APPENDIX E
PROOF OF THE SATISFACTION OF THE SLATER'S QUALIFICATION CONDITION

Let us assume that the conditions of Proposition 3 are met, so that the *RAP* is a convex optimization problem in the $\overrightarrow{RS}$ optimization vector variables. Hence, the Slater's qualification condition requires that there exists at least a *feasible* resource allocation vector that meets the convex constraint in (37b) with the strict inequality (see, for example, [37, Section 5.3]). However, if the feasibility condition in (59) is met with the strict inequality, the vector $\overrightarrow{RS}^{(MAX)}$ of the maximal resources is, by design, feasible, and satisfies the convex constraint in (37b) with the strict inequality. This proves, in turn, that the Slater's qualification condition holds.

REFERENCES

[1] H. C. M. Andrade, B. Gedik, and D. S. Turaga, *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[2] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2013.

[3] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[4] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*. Singapore: Springer, 2018, pp. 103–130.

[5] A. M. Rahmani, P. Liljeberg, J.-S. Preden, and A. Jantsch, *Fog Computing in the Internet of Things*. Cham, Switzerland: Springer, 2018.

[6] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016.

[7] E. Baccarelli, M. Scarpiniti, and A. Momenzadeh, "Fog-supported delay-constrained energy-saving live migration of VMs over multipath TCP/IP 5G connections," *IEEE Access*, vol. 6, pp. 42327–42354, 2018.

[8] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Trans. Cloud Comput.*, to be published.

[9] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.

[10] M. Satyanarayanan *et al.*, "Edge analytics in the Internet of Things," *IEEE Pervasive Comput.*, vol. 14, no. 2, pp. 24–31, Feb. 2015.

[11] A. Yousefpour *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, to be published. doi: 10.1016/j.sysarc.2019.02.009.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[13] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[14] S. Secci, P. Raad, and P. Gallard, "Linking virtual machine mobility to user mobility," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 927–940, Dec. 2016.

[15] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[16] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[17] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," in *Proc. 30th Annu. ACM SIGACT-SIGOPS Symp. Princ. Distrib. Comput. (PODC)*, New York, NY, USA, 2011, pp. 267–276.

[18] D. Dutta, M. Kapralov, I. Post, and R. Shinde, "Embedding paths into trees: VM placement to minimize congestion," in *Proc. 20th Annu. Eur. Conf. Algorithms*, vol. 7501. Berlin, Germany: Springer, 2012, pp. 431–442.

[19] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[20] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. 22th Int. Colloq. Structural Inf. Commun. Complex. (SIROCCO)*. Cham, Switzerland: Springer, Jul. 2015, pp. 104–118.

[21] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, Jul. 2010, pp. 49–62.

[22] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. 31st IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 945–953.

[23] P. Bahl, R. Y. Han, L. E. Li, and M. Satyanarayanan, "Advancing the state of mobile cloud computing," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services (MCS)*, New York, NY, USA, Jun. 2012, pp. 21–28.

[24] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. 6th Int. Conf. Cloud Comput.*, Jun./Jul. 2013, pp. 75–82.

[25] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 74–83, Mar. 2015.

[26] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Mar. 2013.

[27] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "mCloud: A context-aware offloading framework for heterogeneous mobile cloud," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 797–810, Sep./Oct. 2017.

[28] S. Chen, Y. Wang, and M. Pedram, "A semi-Markovian decision process based control method for offloading tasks from mobile devices to the cloud," in *Proc. IEEE Global Commun. Conf. (GLOBCOM)*, New York, NY, USA, Dec. 2013, pp. 2885–2890.

[29] S.-H. Hung, T.-T. Tzeng, G.-D. Wu, and J.-P. Shieh, "A code offloading scheme for big-data processing in android applications," *Softw. Pract. Exper.*, vol. 45, no. 8, pp. 1087–1101, 2015.

[30] T.-Y. Lin, T.-A. Lin, C.-H. Hsu, and C.-T. King, "Context-aware decision engine for mobile cloud offloading," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, New York, NY, USA, Apr. 2013, pp. 111–116.

[31] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, New York, NY, USA, Apr./May 2014, pp. 352–357.

[32] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, New York, NY, USA, Apr./May 2015, pp. 1894–1902.

[33] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, to be published.

[34] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[35] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[36] A. Kumar, D. Manjunath, and J. Kuri, *Communication Networking: An Analytical Approach*. Amsterdam, The Netherlands: Elsevier, 2004.

[37] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Hoboken, NJ, USA: Wiley, 2017.

[38] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "Pricing policy and computational resource provisioning for delay-aware mobile edge computing," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2016, pp. 1–6.
[39] R. Basmadjian and H. de Meer, "Evaluating and modeling power consumption of multi-core processors," in *Proc. 3rd Int. Conf. Future Energy Syst., Where Energy, Comput. Commun. Meet*, New York, NY, USA, May 2012, pp. 1–10.
[40] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 3, pp. 384–398, Sep. 2015.
[41] E. Baccarelli, M. Biagi, R. Bruno, M. Conti, and E. Gregori, "Broadband wireless access networks: A roadmap on emerging trends and standards," in *Broadband Services: Business Models and Technologies for Community Networks*. Hoboken, NJ, USA: Wiley, Oct. 2005, pp. 215–240.
[42] E. Baccarelli and M. Biagi, "Power-allocation policy and optimized design of multiple-antenna systems with imperfect channel estimation," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 136–145, Jan. 2004.
[43] E. Baccarelli, M. Biagi, and C. Pelizzoni, "On the information throughput and optimized power allocation for MIMO wireless systems with imperfect channel estimation," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2335–2347, Jul. 2005.
[44] E. Baccarelli, R. Cusani, and S. Galli, "A novel adaptive receiver with enhanced channel tracking capability for TDMA-based mobile radio communications," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 9, pp. 1630–1639, Dec. 1998.
[45] M. Scarpiniti, E. Baccarelli, P. G. V. Naranjo, and A. Uncini, "Energy performance of heuristics and meta-heuristics for real-time joint resource scaling and consolidation in virtualized networked data centers," *J. Supercomput.*, vol. 74, no. 5, pp. 2161–2198, May 2018.
[46] J.-M. Pierson, *Large-scale Distributed Systems and Energy Efficiency: A Holistic View*. Hoboken, NJ, USA: Wiley, 2015.
[47] Y. Xiao *et al.*, "Modeling energy consumption of data transmission over Wi-Fi," *IEEE Trans. Mobile Comput.*, vol. 13, no. 8, pp. 1760–1773, Aug. 2014.
[48] M. Scarpiniti, E. Baccarelli, and A. Momenzadeh, "VirtFogSim: A parallel toolbox for dynamic energy-delay performance testing and optimization of 5G mobile-fog-cloud virtualized platforms," *Appl. Sci.*, vol. 9, no. 6, p. 1160, Mar. 2019.
[49] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, New York, NY, USA, 2011, pp. 301–314.
[50] E. Baccarelli, M. Biagi, C. Pelizzoni, and N. Cordeschi, "Optimized power allocation for multiantenna systems impaired by multiple access interference and imperfect channel estimation," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 3089–3105, Sep. 2007.
[51] E. Baccarelli and M. Biagi, "Performance and optimized design of space-time codes for MIMO wireless systems with imperfect channel estimates," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2911–2923, Oct. 2004.
[52] E. Baccarelli and R. Cusani, "Recursive Kalman-type optimal estimation and detection of hidden Markov chains," *Signal Process.*, vol. 51, no. 1, pp. 55–64, May 1996.

ENZO BACCARELLI received the Laurea degree in electronic engineering and the Ph.D. degree in communication theory and systems from the Sapienza University of Rome, and the Postdoctorate degree in information theory and applications from the INFOCOM Department, Sapienza University of Rome, in 1995. He is currently a Full Professor in information and communication engineering with the DIET Department, Sapienza University of Rome. His current research focuses on data networks, distributed computing, networked data centers, and Fog computing. From 2005 to 2010, he served as an Associate Editor for the IEEE COMMUNICATIONS LETTERS. He was the national coordinator of several MIUR projects.

MICHELE SCARPINITI (S'06–M'09–SM'15) received the Laurea degree (Hons.) in electrical engineering and the Ph.D. degree in information and communication engineering from the Sapienza University of Rome, Italy, in 2005 and 2009, respectively. Since 2008, he has been an Assistant Professor of Circuit Theory and Multimedia Signal Processing with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome. His current research interests include nonlinear adaptive filters, audio processing and neural networks for signal processing, ICA, and blind signal processing. He is a member of the Intelligent Signal Processing and Multi-Media (ISPAMM) Laboratory, an interdisciplinary research group involved with the DIET Department of Sapienza University of Rome. The ISPAMM research aims at the design and development of innovative methodologies for multimedia processing. He is a member of the Audio Engineering Society (AES) and a member of the Società Italiana Reti Neuroniche (SIREN). He is also on the board of the AES Italian Section.

ALIREZA MOMENZADEH received the B.E. degree in civil engineering from Estahban University, Iran, and the master's degree in structural engineering from the University Technology of Malaysia, in 2015. He is currently a Researcher with the DIET Department, Sapienza University of Rome. His current research activity embraces nonlinear optimization through hybrid methods, Fog computing architectures, and related sensor/actuator-based control applications.