IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 13, 2021, accepted February 9, 2021, date of publication February 12, 2021, date of current version February 23, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3059029*

# Cost-Effective Resource Allocation for Multitier Mobile Edge Computing in 5G Mobile Networks

**EUGEN ŠLAPAK[1], JURAJ GAZDA[1], WEIQIANG GUO[2], (Student Member, IEEE), TARAS MAKSYMYUK[3], (Member, IEEE), AND MISCHA DOHLER[2], (Fellow, IEEE)**

[1]Department of Computers and Informatics, Technical University of Košice, 042 00 Kosice, Slovakia
[2]Centre of Telecommunications Research, Department of Informatics, King's College London, London WC2B 4BG, U.K.
[3]Department of Telecommunications, Lviv Polytechnic National University, 79000 Lviv, Ukraine

Corresponding author: Juraj Gazda (juraj.gazda@tuke.sk)

This work was supported in part by the Slovak Research and Development Agency under Project APVV-18-0214 and Project APVV-15-0055, in part by the Scientific Grant Agency of the Ministry of Education, Science, Research, and Sport of Slovakia under Contract 1/0268/19, and in part by the Ukrainian Government "Designing the novel decentralized mobile network based on blockchain architecture and artificial intelligence for 5G/6G development in Ukraine" under Project 0120U100674.

**ABSTRACT** Mobile edge computing (MEC) is currently one of the key technologies that can facilitate the evolution of the future digitized economy. MEC can provide ubiquitous computational capabilities through the multitier deployment of servers to ensure lower latencies and tighter integration with 5G, the Internet of Things, blockchains and artificial intelligence. In this paper, we propose a new approach to optimizing hardware resource allocation for edge nodes in a multitier MEC hierarchy. In addition to a centralized unit, we consider active antenna units and distributed units equipped with edge nodes of different computational capacities. A parametric Bayesian optimizer is implemented for hardware resource allocation to increase the overall computational capacity of a 5G-based MEC system. Simulation results show that for given budget constraints, the proposed solution outperforms pseudorandom resource allocation in terms of the proportion of computational tasks completed. The achievable gains are in the range of 20-40 %, depending on the task complexity and selected budget threshold.

**INDEX TERMS** Wireless networks, multitier MEC, resource allocation, Bayesian optimization.

## I. INTRODUCTION

With the proliferation of 5G mobile networks, various immersive services such as virtual/augmented reality (VR/AR), ubiquitous computing, online gaming and the Internet of Things (IoT) are penetrating into our daily routines [1]–[4], [5]. Since most of these services require the transmission of massive volumes of data and extremely complex data processing in the cloud, the existing network and cloud infrastructures are likely to be placed under high pressure in the foreseeable future. In recent reports, Cisco has estimated approximately 20 zettabytes of yearly data center traffic by 2021 [6].

Currently, remote servers are typically deployed very far from user equipment devices (UEs) [7], [8]. Thus, requests from UEs are forwarded to a remote server through multiple segments of the transport network infrastructure. Considering the potential tremendous amount of traffic directed from the servers to each UE, the transport network may easily become a bottleneck for future immersive services [9]. Moreover, such remote servers are accessible by billions of UEs, so the probability of congestion is very high for both communication and computing segments. Consequently, the queuing strategy applied for request processing has a direct impact on the performance of mobile cloud computing [10], [11].

Recently, the novel paradigm of mobile edge computing (MEC) has been introduced to alleviate the limitations of mobile cloud computing [12]. The key idea of MEC is to bring the computing infrastructure closer to the end users so that most requests can be processed locally, without being offloaded to remote cloud servers. The necessary local servers can be integrated with the existing mobile network infrastructure, such as base stations and baseband units [13], [14]. MEC provides several essential benefits compared to the conventional cloud computing. First, it reduces round trip latency by placing servers in proximity to end users. Second, by allowing most traffic to be processed locally, MEC reduces the volumes of data that are transmitted far distances through

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

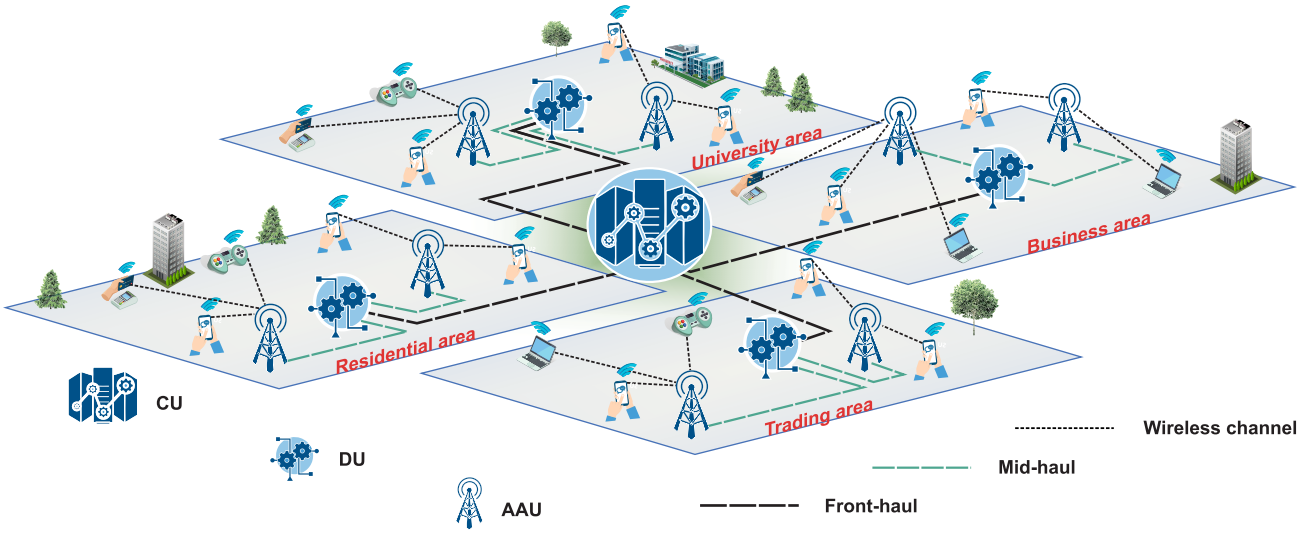



**FIGURE 1.** Proposed MEC architecture, which also provides computational resources at the AAU level.

the transport network [15]. Finally, local edge servers are more suitable for context-oriented services because they can store useful information that is relevant only to a particular area and perform caching of essential content considering the trends in the target area [7], [16]–[18].

Despite all of the benefits of MEC, the deployment of integrated communications and computing infrastructure for MEC is not a trivial task. To date, many research works have pursued the goal of optimal MEC deployment through the proposal of various implementations of game theory, convex optimization and deep learning [7], [19], [20].

To simplify MEC deployment and provide more opportunities for operators, in this paper, we propose a multitier MEC architecture based on an ultradense 5G network [9], as illustrated in Fig. 1. The key conventional assumptions are that each active antenna unit (AAU) is equipped with a low-power MEC server and that each distributed unit (DU) is equipped with a high-power MEC server [9], [21]. However, in this work, all possible approaches are allowed to freely explore the deployment of any amount of computational resources allowed by the global constraints on any type of node to exclude the possibility that any of the presented scenarios may be better suited to configurations that do not obey the above conventions. The two tiers of servers will process most requests from UEs, with the exception of a few ultraheavy requests, which will be forwarded to a centralized unit (CU) in the cloud. Depending on the circumstances, each task should be allocated to the corresponding tier of the proposed MEC system. Despite the large number of research contributions on optimal task allocation in MEC, none of the existing works has considered the optimization of hardware deployment to maximize the computational performance of a multitier MEC system under capital investment constraints.

An additional degree of freedom in our optimization framework is the presence of an exogenous dynamic in the system, which is reflected by the movement of mobile users. In difference to previous works, we take the snapshot of the instantaneous positions of end users to evaluate the load on the MEC servers at each individual simulation run. Consequently, we reflect the overall spatio-temporal computational demand, which allows to optimize the hardware capacity of each tier in the MEC system. In order to simplify an optimization problem, the detailed study of the physical layer properties of 5G network are omitted in this work, considering that it's impact on the service latency is one order of magnitude lower than impact of the computational performance of the MEC system.

Therefore, in this work, we propose an approach for optimizing the computational capacity of a multitier MEC system with given capital investment constraints and given behavior of the end users. Accordingly, the main contributions of this article are as follows:

1) We propose a method for optimal hardware deployment given capital investment constraints for a multitier MEC architecture in which servers are integrated into key elements of the 5G infrastructure, such as the AAUs, DUs and CUs.
2) We simulate the task processing performance in a multitier MEC system with task properties based on real-world measurements of cloud gaming client network activity [22].
3) Our experiments show that with a budget of 600 thousand USD, the proposed optimization method achieves a 41 % increase in the completed task ratio compared to the constrained pseudorandom approach in the 70 FPS cloud gaming scenario and a 29 % increase in the 60 FPS cloud gaming scenario.

The remainder of this paper is organized as follows. In Section II, we provide an overview of recent research activities on MEC optimization. In Section III, we describe the proposed system model and task processing in the multitier MEC system. In Section IV, the proposed

optimization method for multitier MEC deployment is explained. In Section V, simulations of and performance results for the proposed system are discussed. Section VI presents potential future research topics and outlines further possible improvements to MEC, and Section VII concludes the paper.

## II. RELATED WORKS

A traditional cloud computing paradigm, in which all requests are processed at a number of large data centers placed at multiple locations around the world, is inefficient for the dynamic and agile 5G ecosystem. Except for a few rare cases, most mobile services are context-aware services that rely on information that is relevant only to certain local areas. With the rapid increase in data volumes due to rich multimedia applications and the deployment of massive numbers of IoT devices, the transmission of context-aware traffic from multiple areas to the same remote cloud servers is expected to occupy a sufficient share of the total network traffic to negatively impact network efficiency. Therefore, MEC is widely considered an important pillar of the 5G ecosystem to reduce the latency for rich multimedia services and improve the efficiency of context-aware computing.

Whereas in cloud computing, computational power tends to be centralized in large data centers, MEC has the opposite aim of distributing computational power proportionally within target areas, a goal that raises the question of optimality. To date, many research works have focused on the optimization of task allocation for a given MEC system structure.

In [23], the authors mathematically defined and solved the mixed integer dynamic task offloading and scheduling problem. As their objective, they maximized the number of offloaded and processed tasks, under the assumptions that each serving unit can process one task at a time and that the scheduling policy is non-preemptive. This optimization problem is NP-hard, and the authors solved it by implementing logic-based Benders decomposition to reduce its complexity.

Another study [24] has proposed an NP-hard latency-aware workload offloading strategy to minimize the average response time by offloading tasks to a cloud server. The proposed strategy has been shown to yield a significant decrease in average latency compared to a location-aware workload offloading algorithm [25].

An alternative solution was proposed in [26], where the authors performed task partitioning considering latency-sensitive applications and developed several competitive algorithms for evaluating the performance of an MEC server.

In [27], three joint task allocation algorithms were studied: offloading without the consideration of joint data, full offloading and offloading with equal time lengths. The authors defined their objective as minimizing the energy consumption for MEC in a scenario with one MEC-equipped base station and several UEs using AR/VR applications.

It is worth mentioning several related works in which multiobjective optimization has been implemented for the computational and communications segments of MEC systems. Typically, multiobjective optimization problems are very complex (mostly NP-hard); thus, many studies have been conducted to find an effective way to reduce their complexity [23], [27], [28].

In [29] and [30], the objectives were to find the trade-off between the energy consumption and corresponding latency in various offloading scenarios. Specifically, in [29], the time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) schemes were considered for the wireless channels. The researchers evaluated and simulated the energy consumption for both offloading and computing in a multiuser case. On the other hand, in [30], the researchers evaluated the energy consumption for complex tasks and reviewed potential security and privacy improvements for the offloading of tasks to a server.

In [31], the researchers identified that the trade-off problem is a multiobjective optimization problem (MOOP) that can be solved via Pareto optimality, in which the principle is to find the solution such that no other solution improves one of the objectives. To achieve Pareto optimization, they introduced a weighted-sum formulation that can effectively reduce the computational complexity. The considered system consisted of one full-duplex BS equipped with an edge computing server, several MEC users and several mobile service users.

In [32], the authors proposed a framework for jointly minimizing the downlink and uplink transmission power while providing secure data transmission with a given quality of service (QoS). The authors adopted the weighted Tchebycheff method to formulate the resource allocation algorithm as a MOOP.

The joint optimization of computational resources and wireless connectivity was proposed in [33] considering the aspects of energy efficiency and QoS. The authors proposed a device-to-device (D2D) architecture for MEC systems, with the aim of fully utilizing the computational capabilities of the UEs. The authors formulated the NP-hard problems of task and resource allocation for various scenarios in the D2D MEC system. In addition, a convex relaxation-based algorithm was proposed to solve the formulated problems.

In the following sections, we describe the proposed method of hardware deployment optimization for a multitier MEC system considering given financial constraints.

## III. SYSTEM MODEL

The proposed system model for multitier MEC consists of multiple CUs, DUs and AAUs, represented by the notations $A$, $D$, and $C$, respectively; in particular, their quantities are denoted by $N_A$, $N_D$ and $N_C$, respectively [21]. In addition, we denote the numbers of CPUs deployed at these node elements by $N_{\text{CPU\_A}}$, $N_{\text{CPU\_D}}$, and $N_{\text{CPU\_C}}$, respectively. The set of all tasks requested by UEs that exist in the system at any given time step is defined as $T(t)$. In Fig. 2, we show the hierarchical tree structure of the proposed system model, containing a CU at the top of the hierarchy, connected to DUs at the next lower level of the hierarchy, and then AAUs and UEs at even lower levels of the hierarchical tree. Once

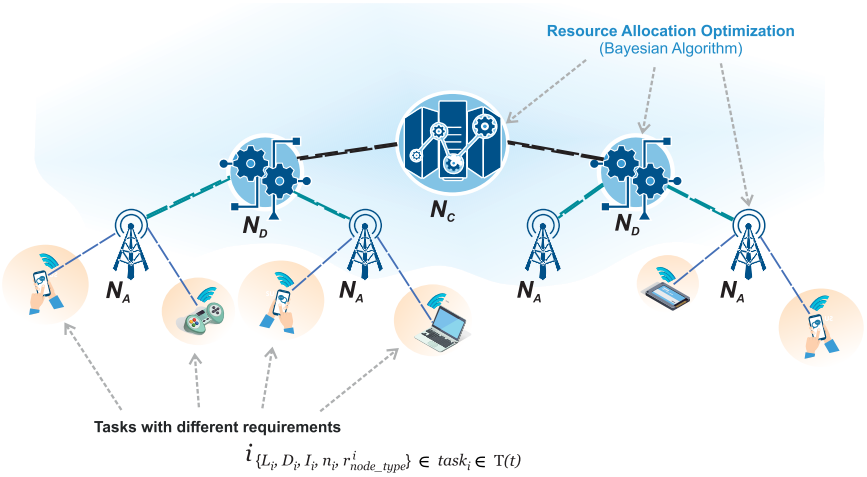



**FIGURE 2.** Simulated hierarchical network tree structure for MEC.

all units have been deployed, the system will automatically connect UEs to the nearest AAUs and AAUs to the nearest DUs; then, all DUs are connected to the only CU present in the simulation. We represent the number of cycles required to process a task as $C_i$, where the index $i$ denotes the index of the $i$-th task. The ratio between the offloaded data size and the number of cycles is defined as the computational intensity $I_i$ [$cycles/bit$], and the size of the $i$-th task is denoted by $D_i$ [34]. Therefore, the required number of computational cycles for the $i$-th task can be calculated as

$$C_i = D_i I_i. \tag{1}$$

For each $\text{task}_i$, there will be a set of parameters, such as $\{L_i, D_i, I_i, n_i, r^i_{\text{node\_type}}\}$. Here, $L_i$ denotes the latency requirements for $\text{task}_i$, and $n_i$ is the ratio between the offloaded data size and the downloaded data size, where the downloaded data size can be calculated as $n_i D_i$. For each task, we use a factor $r^i_{\text{node\_type}} \in \{0, 1\}$ to represent where the task is computed. We assume that a task can be executed at any level of the hierarchy (i.e., by a DU, CU or AAU); thus, for each task, we have three $r^i_{\text{node\_type}}$ factors, expressed as $r^i_A$, $r^i_D$ and $r^i_C$. Analogously, we can introduce

$$r^i_{\text{node\_type}} = \begin{cases} 1 & \text{if } \text{task}_i \in \text{T}(t) \text{ is executed at node\_type} \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

The optimizer tunes the number of nodes of each node type and the number of CPUs deployed on each type of node. Thus, the proposed state of the MEC network is expressed in compact form by the six-tuple

$$s \equiv \langle N_{\text{CPU\_A}}, N_{\text{A}}, N_{\text{CPU\_D}}, N_{\text{D}}, N_{\text{CPU\_C}}, N_{\text{C}} \rangle. \tag{3}$$

Formally, $s \in \mathcal{S}$, where $\mathcal{S}$ is the space of all admissible network states.

**TABLE 1.** Table of notations used.

| Notation | Definition |
|---|---|
| $\text{task}_i$ | task identified by index $i$ |
| $D_i$ | data size of task $i$ |
| $I_i$ | computational intensity of task $i$ |
| $L_i$ | latency requirement of task $i$ |
| $N_{\text{A}}, N_{\text{D}}, N_{\text{C}}$ | numbers of AAU, DU, and CU nodes, respectively |
| $N_{\text{CPU\_A}}, N_{\text{CPU\_D}}, N_{\text{CPU\_C}}$ | numbers of CPUs at respective node types |
| $f_{\text{CPU}}$ | CPU frequency for nodes |
| $r^t_x$ | target level for task $i$ |
| $t_{\text{start},i}$ | starting time step of task $i$ |
| $t_{\text{end},i}$ | ending time step for task $i$ |
| $N_{\text{par\_tasks}}(i, t)$ | number of tasks on the core with $\text{task}_i$ |
| $\Delta t$ | length of a simulation time step |
| $t_{\text{total},i}$ | total time consumed for task $i$ |
| $t_{\text{offload,i}}$ | offloading time for task $i$ |
| $t_{\text{download},i}$ | downloading time for task $i$ |
| $t_{\text{process},i}$ | processing time for task $i$ |
| $t_{\text{F},i}$ | transmission time in fronthaul |
| $t_{\text{M},i}$ | transmission time in midhaul |
| $t_{\text{W},i}$ | transmission time on wireless channel |
| $r_i^{\text{offload}}(t)$ | available data rate on uplink channel for $i$ |
| $r_i^{\text{download}}(t)$ | available data rate on downlink channel for $i$ |
| $B_{\text{W}}^{\text{offload}}$ | bandwidth on uplink wireless channel |
| $p_{\text{W}}^{\text{offload}}$ | transmission power for uplink wireless channel |
| $\sigma^2_{\text{W}}$ | additive white Gaussian noise (AWGN) |
| $f_i(t)$ | computational frequency for task $i$ at time $t$ |
| $P_C$ | price per CPU |
| $P_{\text{serv\_node}}$ | price per server node ($) |
| $P_{\text{rack\_i}}$ | price per server rack or outdoor cabinet ($) |
| $\mathcal{S}$ | space of possible network configurations |

### A. MODELING OF DEPLOYMENT EXPENSES

Since the DUs and CU will be at least partially deployed at already existing sites, only the price for indoor 30U server racks needs to be considered, under the assumption that the necessary supporting infrastructure is already present at the DU and CU sites. Because of the scale of the simulation and the high core counts of the server processors, a single additional rack or outdoor cabinet will be sufficient to accommodate all CPU configurations considered for all simulated nodes. Since it is possible to use a single CPU on a dual-CPU

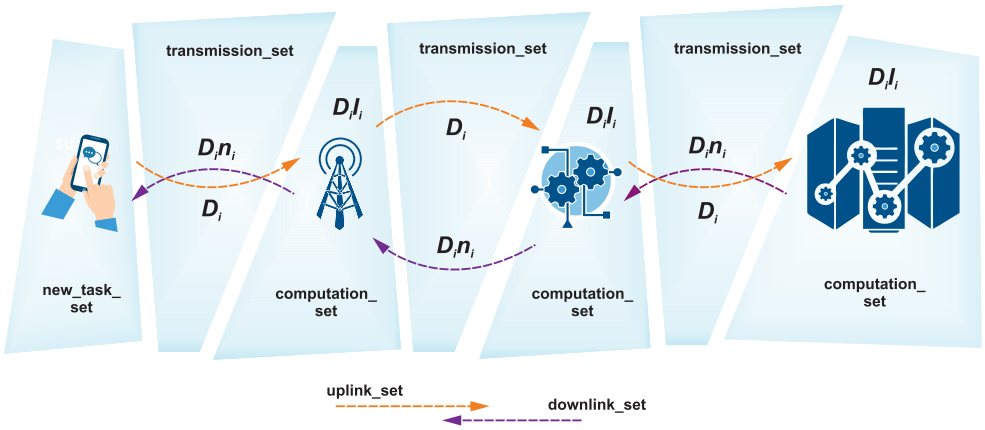



**FIGURE 3.** Existing types of task sets in MEC scenario.

motherboard, the number of server nodes at each network node is computed as $\lceil (N_{CPU\_node\_type}/2) \rceil$. A 30U rack can host 30 server nodes, each with at most 2 CPUs; therefore, the number of racks (coupled with the number of outdoor cabinets for AAUs) needed for one network node can be simply calculated as $\lceil (N_{CPU\_node\_type}/60) \rceil$. The total expenses are formally expressed as shown in Eq. 4, with the constituents given in Eq. 5 - 7.

We express the total expenses as

$$expenses(s) = cpu\_expenses(s) + server\_node\_expenses(s) + rack\_expenses(s) \quad (4)$$

with

$$cpu\_expenses(s) = \sum_{j \in \{A,D,C\}} N_{\text{CPU\_j}} N_j P_{\text{CPU}}, \quad (5)$$

where $P_{CPU}$ is the price of each server CPU. The server node expenses can be expressed as

$$server\_node\_expenses(s) = \sum_{j \in \{A,D,C\}} \lceil N_{\text{CPU\_j}}/2 \rceil N_j P_{\text{serv\_node}}, \quad (6)$$

where $P_{\text{serv\_node\_j}}$ is the price of a single server node and all required hardware is assumed to be provided by the server nodes except the CPUs. Finally, the rack expenses can be determined as follows:

$$rack\_expenses(s) = \sum_{j \in \{A,D,C\}} \lceil N_{\text{CPU\_j}}/60 \rceil N_j P_{\text{rack\_j}}, \quad (7)$$

where $P_{\text{rack\_j}}$ is the price of a single rack or outdoor server cabinet; this price is thus dependent on the type of network node to be hosted.

### B. LATENCY MODELING

Three main contributions to the latency are considered for MEC [33], [35], [36]. These contributions correspond to the three main phases of the task life cycle:

**phase 1 - task offloading**, in which the latency is caused by the transmission of the task to the network node on which it will be executed (the task's target node);

**phase 2 - task execution**, for which the latency is caused by the processing of the task by a particular CPU and is influenced by the computational load imposed by other tasks; and

**phase 3 - task downloading**, in which the task is sent from the target node back to the user to complete its round trip, thus adding the final latency contribution. Each phase is described in more detail later in this subsection.

Formally, we express the total latency for task$_i$ as

$$t_{\text{total},i} = t_{\text{offload},i} + t_{\text{process},i} + t_{\text{download},i}. \quad (8)$$

In the system model, to simulate a large number of simultaneously interacting network elements and tasks, sets with equal division of simultaneously shared resources among the tasks, that are parts of the processed sets, is introduced. Each network node hosts three types of such sets to model the division of the bandwidth needed for uplink and downlink and the division of computational resources among the set elements: **uplink_set**, **downlink_set** and **computation_set**.

#### 1) PHASE 1: TASK OFFLOADING

The tasks are immediately sorted into **uplink_set** once they are generated by the UEs, as illustrated in Fig. 3. **uplink_set** simulates the transmission of a task to a node at a higher hierarchical level, with all parallel tasks in the same set sharing the transmission link resources. Similarly, **downlink_set** represents transmission from the MEC nodes towards the UEs.

Then, the offloading time $t_{\text{offload},i}$ can be expressed as

$$t_{\text{offload},i} = t_{\text{w},i} + t_{\text{F},i} + t_{\text{M},i}, \quad (9)$$

where $t_{\text{w},i}$ denotes the time taken for wireless transmission between the UE and the AAU, $t_{\text{F},i}$ is the fronthaul transmission time between the AAU and the DU, and $t_{\text{M},i}$ is the midhaul transmission time between the DU and the CU.

For the wireless link $t_{\text{w},i}$, let $h_{\text{W}}$ denote the channel power gain from the UE to the AAU for offloading; then, the desired offloading rate is given by

$$r_i^{\text{offload}} = B_{\text{W}}^{\text{offload}} \log_2 \left( 1 + \frac{p_{\text{W}}^{\text{offload}} h_{\text{W}}}{\sigma_{\text{W}}^2} \right), \quad (10)$$

where $B_{\mathrm{W}}^{\mathrm{offload}}$ (in Hz) denotes the transmission bandwidth, $p_{\mathrm{W}}^{\mathrm{offload}}$ is the transmission power for offloading tasks to the AAU, and $\sigma_{\mathrm{W}}^2$ is the additive white Gaussian noise (AWGN) at the AAU [33].

In a multiuser scenario, the data rate is given by

$$r_i^{\mathrm{offload}}(t) = \frac{r_{\mathrm{W}}^{\mathrm{offload}}}{N_i(t)}. \tag{11}$$

Therefore, $r_i^{\mathrm{offload}}(t)$ is a time-varying quantity due to the varying number of parallel tasks $N_i(t)$. A task can be considered transferred when Eq. 12 is satisfied:

$$D_i \leqslant \sum_{t_{\mathrm{start},i}}^{t_{\mathrm{end},i}} r_i^{\mathrm{offload}}(t)\Delta t. \tag{12}$$

Finally, the transmission time for task$_i$ is computed as

$$t_{\mathrm{w},i} = (t_{\mathrm{end},i} - t_{\mathrm{start},i})\Delta t. \tag{13}$$

Regarding the wired links, the fronthaul and midhaul transmission times can be calculated in the same way since both types of connections use optical fibers. To avoid repetition, the notation used below does not distinguish between uplink and downlink. For example, the fronthaul latency is given as $l_{\mathrm{F}} = \frac{L_{\mathrm{F}}}{c}n$, where $L_{\mathrm{F}}$ is the length of the fronthaul optical fiber, $c$ is the speed of light, and $n$ is the refractive index of the glass in the cable. Analogously, the fronthaul offloading time is given as $t_{\mathrm{F},i} = \frac{D_i}{C_{\mathrm{F},i}} + l_{\mathrm{F}}$, where $C_{\mathrm{F},i}$ is the capacity of the fronthaul link on which task$_i$, with message (request or response) size $D_i$, is present.

Similarly, the time taken for offloading $D_i$ through a midhaul connection is given as $t_{\mathrm{M},i} = \frac{D_i}{C_{\mathrm{M},i}} + l_{\mathrm{M}}$.

#### 2) PHASE 2: TASK EXECUTION

In the proposed discrete-event simulation, all tasks present on any particular core in a given time step are required to share the core instruction cycles available per unit of time. The execution time for any task will depend on the dynamics of any other tasks present on the CPU core in each individual time step. Because the tasks considered in this work are identical, equal division of the computational resources among the tasks present on a given core satisfies the fairness requirement, and for a specific task$_i$, the time-varying frequency it is assigned in time slot $t$ can be calculated as

$$f_i(t) = \frac{f_{\mathrm{CPU}}}{N_{\mathrm{par\_tasks}}(i,t)}, \tag{14}$$

where $N_{\mathrm{par\_tasks}}(i,t)$ denotes the number of parallel tasks being processed on the same CPU core as task$_i$. However, several CPUs will be deployed on the same node, with each containing multiple cores; thus, the total number of tasks on one node in a specific time slot $t$ is denoted by $N_C(t)$, and $N_{\mathrm{par\_tasks}}(t)$ can be computed as

$$N_{\mathrm{par\_tasks}}(i,t) = \frac{N_{\mathrm{tasks}}(\mathrm{node,\ i})}{N_{\mathrm{cores}}(\mathrm{node\_type})}. \tag{15}$$

There is a limit on how many parallel tasks a single CPU can support, but this number is extremely large considering that one CPU can switch between tasks very quickly; thus, this limit can be effectively neglected. The computational set therefore depends on the number of CPUs deployed at each node, as all CPUs will share all tasks evenly.

Once a task is transmitted to its target node, it will be sorted into **computation_set** and start to be processed. The analytical model used to simulate the process at the nodes has been described in previous sections. In the actual simulation, the number of calculation cycles elapsed and the number of parallel tasks are checked in every simulation time step for each task. A task is considered completed when the number of elapsed cycles is larger than or equal to the task's requirement, as follows:

$$D_i I_i \leqslant \sum_{t_{\mathrm{start},i}}^{t_{\mathrm{end},i}} f_i(t)\Delta t, \tag{16}$$

where $t_{\mathrm{start,i}}$ and $t_{\mathrm{end,i}}$ are the time slots in which the task begins and ends processing, respectively; thus, the actual processing time for task $i$ can be calculated as

$$t_{\mathrm{process},i} = (t_{\mathrm{end},i} - t_{\mathrm{start},i})\Delta t. \tag{17}$$

#### 3) PHASE 3: TASK DOWNLOADING

When the execution of a task is complete, the result will be moved to **downlink_set** to be transmitted back to the UE. The mechanism of interactions for the downlink transmission set is the same as that for the uplink transmission set: the size of the data to be transferred is calculated based on the number of parallel users/tasks in each time step, and a task is considered transmitted once the delivered data size is equal to or larger than the task size, as in Eq. 12.

There is no ambiguity in determining the task routing when moving a task up the network tree hierarchy to be computed by the target node, as each node has only a single parent; thus, to route the task, it is sufficient to determine only the target level index. On the other hand, when moving a computation result down the hierarchy back to the UE that requested the computation, the path back through the network to this UE must be unambiguously determined. Since IP addressing and routing are not implemented in our model, a simplified mechanism using a stack data structure local to each task is implemented to record all nodes visited by a task on its way to the target node.

## IV. SYSTEM OPTIMIZATION

### A. OBJECTIVE FUNCTION

The main objective in this research is to maximize the performance of MEC deployment within a limited budget. Therefore, as a measure of performance, the ratio of the number of failed tasks ($N_{\mathrm{failed}}$) to the total number of generated tasks ($N_{\mathrm{T}}$) is considered. The failed tasks are defined as those that exceed their latency requirements $L_i$ and are consequently discarded for network congestion management. In addition, we need to ensure that the operator's total expenses do not exceed a given budget threshold $B_{thresh}$. This assumption is integrated into the optimization algorithm such that any

configuration choice made by the optimizer is guaranteed to satisfy the budget constraints. Thus, the objective function can be expressed as

$$\begin{aligned} &\underset{s}{\text{minimize}} f(s,u) = \frac{N_{\text{failed}}(s,u)}{N_{\text{T}}} \\ &\text{subject to} \quad expenses(s) < B_{\text{thresh}} \\ &N_{\text{A}}, N_{\text{D}}, N_{\text{C}} > 0, \quad N_{\text{A}}, N_{\text{D}}, N_{\text{C}} \in s \\ &N_{\text{CPU\_A}}, N_{\text{CPU\_D}}, N_{\text{CPU\_C}} > 0, \\ &N_{\text{CPU\_A}}, N_{\text{CPU\_D}}, N_{\text{CPU\_C}} \in s \\ &N_{\text{T}} > 0 \end{aligned} \tag{18}$$

where $s$ is the tuple describing the CPU and node counts, as defined in Eq. (3). In addition to the optimized $s$, $N_{\text{failed}}(s,u)$ also depends on the MEC parameters represented by the common symbol $u$. These parameters, which are related to the task complexity, the task size, the connection throughput, the number of users present in the simulation, the frequency of the CPUs used, etc., are not subject to the optimization process.

Again, the value of $expenses(s)$ is guaranteed to not exceed $B_{\text{thresh}}$; thus, $f(s,u)$ is not defined for values of $s$ such that $B_{\text{thresh}}$ would be exceeded.

### B. BAYESIAN OPTIMIZATION

The Bayesian tree-structured parzen estimator (TPE) optimizer was first proposed in [37] and by its nature it resembles heuristic-based stochastic optimization in variable landscape [38]. We need to appreciate the fact that the optimizer performs well especially when the computation of the cost objective function is expensive. In our case, we can claim that the model under consideration is complex, consisting of the large set of AAUs and DUs. Here the overall network architecture (number of AAUs and DUs) and corresponding CPU deployment needs to be optimized, considering the budget constraints of the operator. Evaluation of the objective function for each tuned configuration would require large time complexity, as calculating even a single $f(s,u)$ value for a simulation model scenario cycle requires a rather costly computational effort. Thus, in general we can claim that; *a) the problem is complex and non-linear* and *b) the evaluation of each instance of objective function is time expensive.* Typical solutions including dealing with the problem as the constraint satisfaction problem would be difficult (tree search algorithms with some tree pre-processing (forward checking, arc consistency, etc). As the literature suggest [39], in these cases, local search algorithms (hill climbing, simulated annealing) but more importantly algorithms solving the more difficult problem of global search, are of prominent interest. By its nature, we can assign Bayesian TPE optimizer to the latter group and thus we found it extremely interesting. Specifically, a Bayesian TPE optimizer is adopted for this research to reduce the number of objective function evaluations. Compared to blind search strategies, this algorithm includes a mechanism to ensure that historical objective function values are widely used rather than being forgotten during the optimization process. Finally referring to [39], provided we ensure that the TPE optimizer iterates over large number of steps, the algorithm will find a global optimum with probability approaching 1. Based on our large numerical experiments we can claim with high certainty that at the end, the solution proposed by Bayesian TPE optimizer could be declared as the global optimum or the point very close to the global optimum in the parametric search space (almost optimal). Thus, it could serve as the good alternative to the well-known analytic representatives with precise performance prediction capabilities reached at the cost of severe model relaxation.

The Bayesian optimizer, or more specifically, its Python-based implementation provided in the Hyperopt library [40], belongs to the class of sequential model-based global optimization (SMBO) algorithms, which consecutively create a model of the objective function. Strategically, it selects candidate parameter changes using a surrogate function regression model based on the TPE. In the following, we will continue to discuss optimization mostly in terms of probabilistic models. Examples of probability densities are generally denoted by $p(\ldots)$. While the expected improvement for Bayesian optimization based on a Gaussian process model is calculated via direct modeling of $p\left(f(s,u)|s\right)$, the TPE approach involves calculating $p\left(f(s,u)|s\right)$ from $p\left(s|f(s,u)\right)$, $p\left(f(s,u)\right)$ and $p\left(s\right)$. As can be readily seen, $p\left(f(s,u)|s\right)$ is the posterior distribution.

It is now feasible to apply parameterization to write the conditional distribution function $p\left(s|f(s,u)\right)$ as follows:

$$p\left(s|f(s,u)\right) = \begin{cases} l\left(s\right) & \text{if } f(s,u) < f^* \\ g\left(s\right) & \text{if } f(s,u) \geq f^* \end{cases}, \tag{19}$$

where $f^*$ is a threshold parameter that defines the successive sampling process and $l(s)$ and $g(s)$ are distributions that partially describe $p(s|f(s,u))$. Candidate parameters for objective function evaluations are chosen using the *expectation value of the improvement function*:

$$\begin{aligned} EI_{f^*}(s) &= -\sum_{\substack{s \in \mathcal{S} \\ f(s,u) < f^*}} \left(f^* - f(s,u)\right) p\left(f(s,u)|s\right) \\ &\propto -\left(\gamma + \frac{g\left(s\right)}{l\left(s\right)}\left(1-\gamma\right)\right)^{-1}. \end{aligned} \tag{20}$$

Here, $f^*$ is conditioned on the optimizer's hyparameter $\gamma$ such that $p(f(s,u) < f^*) = \mathbf{1}_{f(s,u)<f^*} = \gamma$.

As a result, in each optimization step, we have a set consisting of all previously calculated objective function values for different vector parameters such that $f(s,u)$ is less than $f^*$. In mathematical terms, we can conclude that in the situations described above, we are led to a *low-level set* that is adopted to model the probability density $l(s)$ in Eq. (19). The complementary subset, which satisfies $f(s,u) \geq f^*$, is used to define the probability density $g(s)$. Both $g(s)$ and $l(s)$ are obtained from collected samples (objective function evaluations for parameter vectors) using *Parzen estimation*.

As the name suggests, $EI_{f^*}(s)$ balances the exploration of regions with high and low chances of improving the objective function based on future improvements.

In each optimization step, first, a set of potential parameter values is chosen based on $l(s)$. Then, the evaluation candidate is selected using Eq. (20). The progressive changes to the model in terms of the objective function estimates based on the obtained measurements are reflected in the gradual changes in the $EI_{...}$ values for the same inputs.

Pseudocode in Alg. 1 describes the optimization process with $N_{\text{iterations}}$, where $N_{\text{iterations}} = 1000$. Before the algorithm is able to suggest suitable candidate parameters, it needs to add sufficient amount of measurements (objective function values, paired with parameters used for their evaluation) to a set of measured samples, by running warmup iterations with random parameter values. This is needed to build sufficient approximation of conditional distribution shown in Eq. 19 in order to calculate $EI$ for candidate parameters according to Eq. 20.

**Algorithm 1** Optimization Algorithm

```
sample_set ← ∅
for warmup_iteration ← 0 to N_warmup_iterations by 1 do
    s ← pseudorandom(B_thresh)
    sample_set.insert(f(s, u), s)

for iteration ← 0 to N_iterations by 1 do
    max_EI ← 0
    candidate_set ← get_candidate_set(sample_set)

    max_EI ← 0
    foreach candidate s_i ∈ candidate_set do
        If EI(s_i, sample_set) > max_EI max_EI ←
        EI(s_i, sample_set) s_best_EI ← s_i
    sample_set.insert(f(s_best_EI, u), s_best_EI)
```

After this exploration phase, multiple candidate parameters are repeatedly chosen, but only the parameters with best $EI$ are used to evaluate computationally expensive function $f(s, u)$. When the algorithm terminates, the *sample_set* will contain all measurements along with measurement containing $f(s, u)$ with lowest found value (lowest ratio of uncompleted tasks) and tuned parameters $s$ used to achieve this value.

### C. PSEUDORANDOM RESOURCE ALLOCATION

We have decided to use the pseudorandom resource allocation as the baseline for the performance improvement delivered by Bayesian TPE optimizer. Expenses available for pseudorandom resource allocation at each MEC network layer are assigned to individual network levels so that their sum is equal to budget threshold as follows:

$$B_{\text{A\_thresh}} + B_{\text{D\_thresh}} + B_{\text{C\_thresh}} = B_{\text{thresh}}, \tag{21}$$

and at the same time constraint $B_{\text{j}} >= expenses_j(s\_min_j, u)$ must be met, because each network layer should contain at least a single node with at least a single processor. Thus, $s\_min_j$ equals $\text{s}_\text{j}$ such that $N_{j\_min} = 1, N_{j\_CPU\_min} = 1$, $N_{j\_min}, N_{j\_CPU\_min} \in s\_min_j$.

Once randomly selected amount of the total budget is assigned to particular MEC level, the largest possible number of CPUs at a particular MEC network level is given as

$$N_{\text{j\_CPU\_max}} = \arg\max_{N_{\text{j\_CPU}}}(expenses(\text{s}_\text{j})), \quad N_{\text{j\_min}}, N_{\text{j\_CPU}} \in s_j \quad \text{subject to} \quad (s_j) < B_{\text{j\_thresh}}. \tag{22}$$

When $N_{\text{j\_CPU\_max}}$ is known, specific value $N_{\text{j\_CPU\_spec}} = random(1..N_{\text{j\_CPU\_max}})$ is chosen. The variable $N_{\text{j\_CPU\_spec}}$ specifies the random number of CPUs that is guaranteed to stay within the budget for $j$-th network level if deployed within at least one, or more nodes. Specific number of nodes that will be deployed at given level $N_{\text{j\_spec}}$ is determined as the largest number of nodes, with each node containing $N_{\text{j\_CPU\_spec}}$ CPUs, that can be purchased without exceeding $B_{\text{j\_thresh}}$, as shown in Eq. 23

$$N_{\text{j\_spec}} = \lfloor B_{\text{j\_thresh}} / expenses(s_j) \rfloor, \quad N_{\text{j\_min}}, N_{\text{j\_CPU\_spec}} \in \text{s}_\text{j}. \tag{23}$$

This pseudorandom allocation scheme ensures that expenses approach $B_{\text{j\_thresh}}$ as much as possible so that financial resources assigned to given level are not unnecessarily wasted.

## V. METHODOLOGY FOR MEASURING SIMULATION PERFORMANCE

Two scenarios differing in the latency tolerated by the UEs are analyzed, with the tolerated latency also dictating the rate at which tasks are generated. In the first scenario, the deadline is set to approximately 16.667 ms, corresponding to rendering at 60 frames per second (FPS) provided by the MEC system. The deadline value for the second scenario is set to approximately 14.286 ms, corresponding to 70 FPS rendering. Other properties of the tasks in both scenarios, including the number of cycles needed to compute each task and the sizes of the requests and responses between the user and target nodes, are identical.

The measurement iterations for evaluating optimized and random network configurations correspond to $n$ budget steps from *min_budget* to *max_budget_limit*. For the $n$-th budget step, $B_{\text{thresh},n} = B_{\text{thresh},(n-1)} + budget\_step$, where the initial threshold is $B_{\text{thresh},0} = min\_budget$, and $B_{\text{thresh},n} <= max\_budget$ in each step.

Additionally, there is an inner loop of the optimization algorithm - for each budget step, $trial\_count = 1000$ simulations are run repeatedly in a Monte Carlo manner to obtain aggregate metric measurements. The parameters of the random simulation runs are averaged over all runs, but only configurations that do not exceed the budget are allowed. A large *trial_count* ensures a value that is close to the expected value due to the law of large numbers for average random runs and thus ensures that the optimization results are close to optimal.

The other task properties are close to those of cloud gaming workloads. Specifically, each user request message is set to 1024 bits in size and contains user input that controls the server-resident process. While the size of the job to be computed is not relevant to network transmission because it resides on the server, the number of cycles required to complete the job determines the duration of the server's delay before the response message is sent. The number of cycles needed to process a single task (render a single frame) in our simulation is $2.31 * 10^7$ cycles. In addition, the size of the response from the target server is set to 100000 bits per message, corresponding to the approximate amount of data needed to send a single compressed frame that is part of a video stream (a sequence of consecutive server responses). To determine these values, approximate request and response sizes were obtained empirically while interacting with the Nvidia Now cloud gaming service, using Wireshark to record the network activity [22]. Finally, to simulate the simultaneous multithreading speedup of the simulated Intel processors, the core speed of each simulated processor is increased by approximately 20% [41].

The calculation of the expenses depends strictly on the constant prices of the network node components and the chosen numbers of processors per node type, which determine all other accompanying resources and infrastructure. All of the following prices are estimated average market prices at the time of the writing of this work. The price of a single 18-core CPU with a frequency of 2.4 GHz and an L3 cache size of 45 MB is approximately 2400 USD. Each CPU is installed on a server node, with each server node hosting at most 2 CPUs due to motherboard limitations. The price of a single server node with a dual-CPU motherboard including 196 GB of RAM, a 4 TB hard drive and a 250 GB SSD drive, but excluding the cost of the CPUs, is also approximately 2400 USD. The price of a 30U server cabinet intended for external placement, as needed to host the hardware for AAUs, is 2700 USD, excluding other costs needed to provide the cabinet with networking capabilities and electrical power, which will probably increase the cost (along with the cost of the required deployment procedures). Thus, the lower bound for the deployment of an exterior cabinet, which is also used for expense calculations in our model, is 3500 USD.

The pseudocode given in Alg. 2 formally describes the measurement methodology applied throughout the simulation study, and Tab. 2 provides a list of simulation parameters and corresponding values.

**Algorithm 2** Measurement Methodology Algorithm

```
/* Overall budget limits in USD ($)    */
max_budget_limit ← 1600000
min_budget ← 400000
budget_step ← 100000

B_thresh ← min_budget

while (B_thresh <= max_budget) do
    mc ← max_conf_cpus(B_thresh)
    mn ← max_conf_nodecounts(B_thresh)
    max_conf ← mc, mn

    if random_trials == TRUE then
        i ← 0
            metrics_tr ← trials(trial_count, max_conf)
            /* get average                  */
        metrics ← metrics_tr/trial_count
    else
        /* return trial with lowest obj.
           func. metric vector component
           value                            */
        metrics ← optimization(trial_count, max_conf)
    B_thresh ← B_thresh + budget_step

function trials (trial_count, max_conf):
    while i < trial_count do
        conf ← rand_conf(max_conf)
            /* this is a vector of multiple
            metrics, including the objective
            function                        */
        metrics_tr ← metrics + network_eval(conf)
        i ← i + 1
    return metrics_tr
```

## VI. RESULTS

In this section, we present simulation results for the proposed system model. We compare the performance of the proposed optimum search algorithm with that of conventional pseudo-random hardware resource allocation.

Ability of optimization algorithm to find global optimum is illustrated in Fig. 4, showing that after 1000 optimization iterations used in our experiments, the algorithm achieves lowest possible value of objective function for higher budgets.

Two scenarios with different user quality requirements are considered, corresponding to refresh rates of 60 FPS and 70 FPS. The actual costs for the solutions found by the algorithms are illustrated in Fig. 5, showing that both approaches, by design, closely approach the budget threshold but never exceed it.

In general, the optimum search algorithm outperforms pseudorandom resource allocation regardless of the refresh rate and budget constraints. However, the greatest advantage of the proposed algorithm is observed for lower budgets. With an increasing budget, both randomly deployed network configurations and optimized network configurations show acceptable performance, regardless of the quality requirements. As expected, both algorithms perform better for the 60 FPS scenario because of the lower requirements.

Fig. 6 shows the trade-off between the total budget allocated for MEC network configuration and the corresponding performance. In the lower budget range, even optimized configurations cannot provide sufficient computational power

**TABLE 2. Table of simulation settings.**

| Parameter | Value |
|---|---|
| $N_{\text{UE}}$ (number of UEs) | 150 |
| UE uplink throughput | 25 mbit/s |
| UE downlink throughput | 100 mbit/s |
| AAU, DU uplink throughput | 40 gbit/s |
| AAU, DU downlink throughput | 40 gbit/s |
| $N_{\text{iterations}}$ (trials per $B_{\text{thresh}}$) | 1000 |
| Simulated trial duration | 6 s |
| $C_i = D_i * I_i$ | $2.31 * 10^7$ cycles |
| Request size | 1024 bit |
| Response size | 100 kbit |
| $L_i$ for 60 FPS scenario | ~ 16.667 ms |
| $L_i$ for 70 FPS scenario | ~ 14.286 ms |
| $f_{\text{CPU}}$ | 2.4 GHz |
| Number of cores per CPU | 18 |
| $\Delta t$ | 10 ms |
| $P_C$ | $2400 |
| $P_{\text{serv\_node}}$ | $3400 |
| $P_{\text{rack\_A}}$ | $3500 |
| $P_{\text{rack\_D}}$, $P_{\text{rack\_C}}$ | $1500 |
| Rack capacity | 30 server nodes |
| Network node placement | Binomial point process |
| UE request activity | Continuous, clocked by $L_i$ |
| Transmission and CPU scheduling | Round-robin |

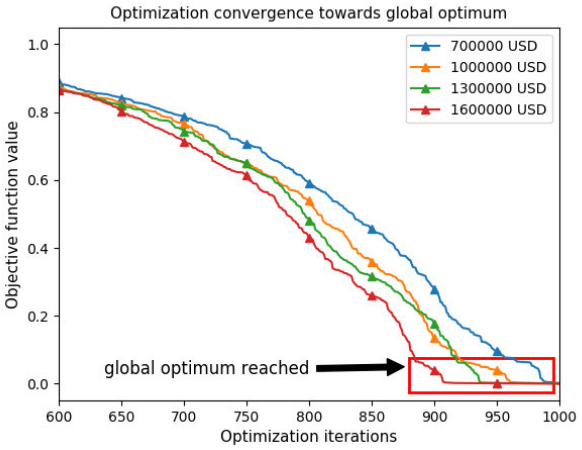



**FIGURE 4. Objective function values for a subset (400 best out of 1000) of optimization iterations, for various deployment budgets. Note that iterations were ordered according to their resulting objective function values in this plot, for better clarity of optimization trends. Bayesian optimizer is not guaranteed to monotonously decrease objective function values with increasing optimization iterations, and the set of parameters resulting in best objective function value can be found in any iteration.**

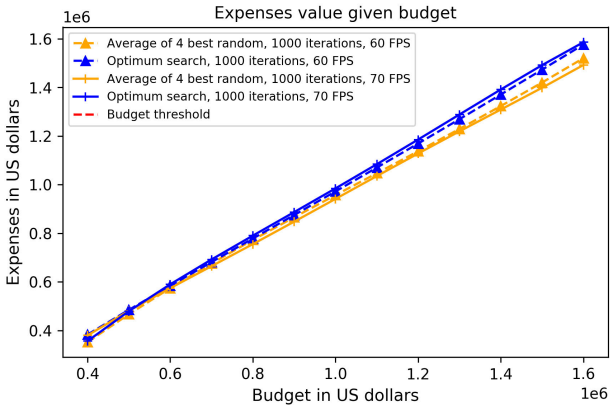



**FIGURE 5. Expenses for MEC network configurations found by the optimization algorithm based on 1000 simulation runs and the average of the 4 best results obtained from 1000 budget-constrained random runs.**

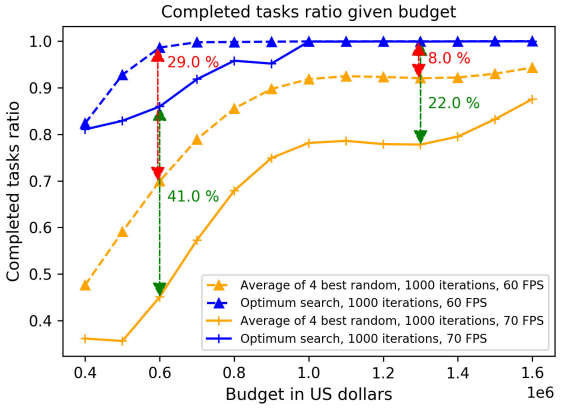



**FIGURE 6. Proportion of tasks completed using the network configuration found by the optimization algorithm based on 1000 simulation runs and the average of the 4 best results obtained from 1000 budget-constrained random runs. The red arrows illustrate the percent increases in this metric obtained through optimization for the 60 FPS scenario, and green arrows illustrate the same for the 70 FPS scenario.**

for certain numbers of users and their tasks. On the other hand, the randomly deployed systems cannot satisfy all tasks for any simulated budget, although the completed task ratio increases with higher budgets. For a budget of 600 thousand USD, optimization enables a 29 % increase in the number of completed tasks for the less demanding 60 FPS scenario. For the higher computational requirements of the 70 FPS scenario and the same budget value constraint, the optimizer is able to achieve a 41 % improvement in the task completion metric compared to the constrained pseudorandom approach. For optimization in the 60 FPS scenario, all tasks are completed when the budget is raised to approximately 700,000 USD, and for the 70 FPS tasks, all tasks are completed when the budget is at least 1,000,000 USD. For a budget of 1.3 million USD, the optimization approach achieves improvements over the random approach of 8 % and 22 % for the 60 FPS and 70 FPS scenarios, respectively. We can observe that relative improvement enabled by optimization compared to the constrained pseudorandom approach increases with increasing difficulty of the simulation runs in terms of both budget and deadline constraints, as reflected by a decreasing number of relatively successful solutions in the search space. Larger budgets and lower FPS rates decrease the challenge for the random search method, resulting in smaller differences between the random search results and the optimized results for easier computational loads.

It is evident that the configurations found by the optimization algorithm cost much less than pseudorandomly chosen configurations with the same performance when the budgets are higher because the pseudorandom approach yields diminishing returns on investments as the budget increases.

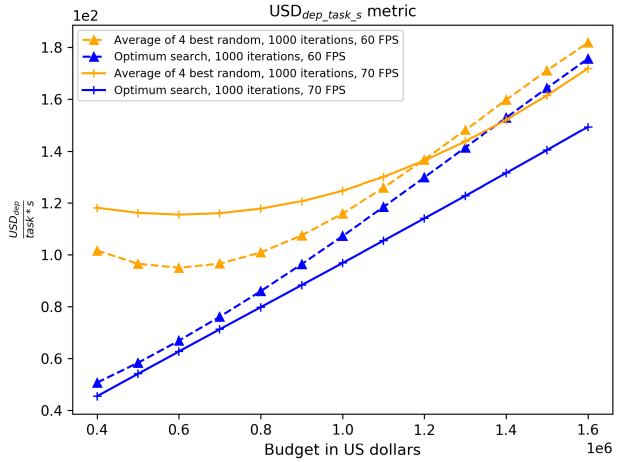



**FIGURE 7.** Deployment expenses per completed task per second for MEC network configurations found by optimization algorithm using 1000 simulation runs and average of 4 best results obtained from 1000 budget-constrained random runs.

Therefore, the optimized configurations are able to provide better performance when the budget is low and cost less when the budget is high.

Apart from overall deployment costs, it is also useful to calculate the deployment expenses per single completed task delivered for both considered resource allocation approaches and scenarios. As computational resources and their overall costs influence how many tasks with given parameters can be completed by the MEC network per second, we can define the unit of the mentioned metric as $[USD_{\text{dep}} task^{-1} s^{-1}]$ where $USD_{\text{dep}}$ is the amount of dollars that were spent strictly on MEC network deployment, without inclusion of any operational expenses (OPEX).

The metric can be calculated as

$$USD_{\text{dep\_task\_s}} = \frac{USD_{\text{dep}}}{N_{\text{UE}} * FPS * \frac{N_{\text{failed}}(s,u)}{N_{\text{T}}}}, \tag{24}$$

where $N_{\text{UE}}$ is the number of connected UEs and $\frac{N_{\text{failed}}(s,u)}{N_{\text{T}}}$ is the ratio of completed tasks. Single generated frame corresponds to single task, so that $N_{\text{UE}} * FPS$ is the total number of tasks generated in the network per second.

The metric is plotted in Fig. 7 as a function of the budget. Here we can see that for lower budgets optimization approach outperforms random approaches in both scenarios, with difference in performance of pseudorandom runs decreasing and then stabilizing with higher budgets. Improvement of $USD_{\text{dep\_task\_s}}$ is more significant in case of optimization for 70 FPS scenario.

Finally, Fig. 8 shows the numbers of CPUs deployed at the CU level (subfigure **(a)** and AAU level (subfigure **(b)** for different budget ranges and scenarios. The results for deployment of CPUs at CU show that the random network configuration algorithm simply increases the CPU count at CU as the available budget increases. By contrast, the optimization algorithm actually deploys a smaller number of CPUs to the CU because it identifies that this parameter is of lower importance and that the saved expenses can be more effectively used to improve other network parameters, like notable increase of cumulative CPU count of nodes at AAU level compared to random runs for higher budgets. DU level was not included in this comparison, as there is not a clearly visible pattern for deployment of CPUs at this level, like the difference between CU and AAU resource allocation.

## VII. FUTURE RESEARCH

### A. MULTI-OPERATOR FRAMEWORK FOR SERVICE PRICING OF THE END-USERS

Further step towards the optimization of the MEC would be the inclusion of the multi-operator framework, where the MEC operators compete to attract the end-users demand. The operators with different sizes and configurations would need to optimize their network architecture taking into account spatial/temporal characteristics of the region and the presence of the competitors. Such a scenario would require a fine grained MEC resource sharing on the level of virtual machines, which can be rented by operators. In such a scenario, we expect to observe service price competition resulting in possible emergence of cartels, or alternatively the price war, from which the end-user can benefit from. In addition, further advancements in modelling of the end-user behaviour with different services and requirements would be welcomed in this area. Introduction of end-user oriented metric such as US dollar/task (in terms of OPEX - not to be confused with metric presented earlier that was derived from deployment expenses) would bring us additional insight to the behaviour and sustainability of the multi-operator framework in the presence of various edge demand services (video streaming, data, augmented reality), co-existing in such a complex techno-economic ecosystem.

### B. SPECIFIC IMPROVEMENTS FOR ARTIFICIAL INTELLIGENCE WORKLOADS

One topic neglected in this work is the role of artificial intelligence (AI) acceleration in MEC networks in the near future [42], [43]. MEC is an essential technology for facilitating the widespread development of 5G networking and beyond. AI workloads can benefit from specialized hardware dedicated to tasks of this kind - from accelerators using decreased floating point precision coupled with the single instruction multiple data (SIMD) architecture with some specific improvements, to paradigm-shifting neuromorphic dataflow chips such as the IBM TrueNorth. AI accelerators do not necessarily need to accelerate the training process, as often only inference using a pretrained model is required after deployment, thus enabling additional savings by excluding unneeded hardware functionality.

As shown by OpenAI [44], the history of AI research milestones can be divided into two eras. The first era lasted from the dawn of AI research until 2012 and was characterized by increases in computational requirements per model that were consistent with Moore's law. Since 2012, the field

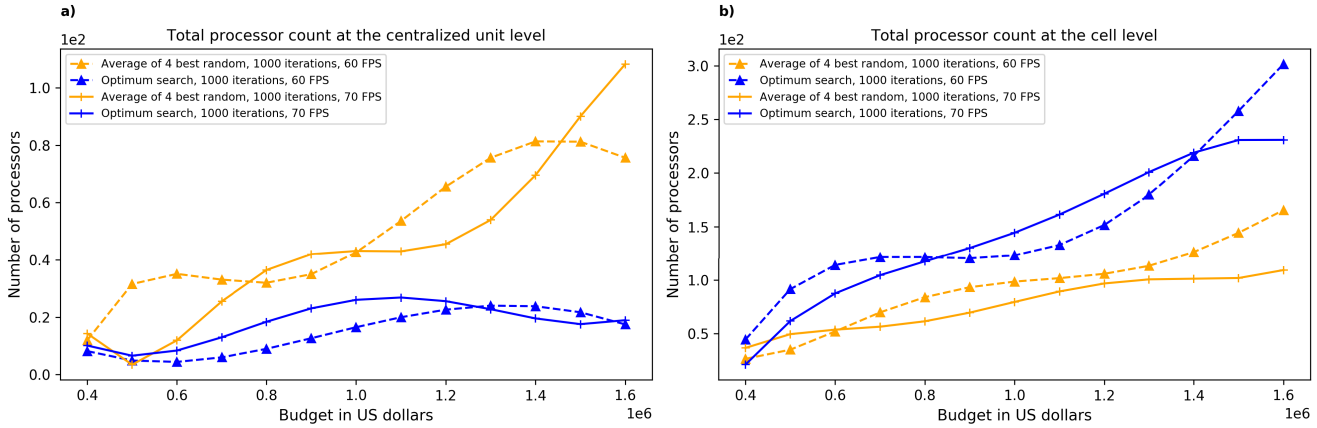



**FIGURE 8.** Overall CPU count assigned to nodes at the CU and AAU MEC newtork level by the optimization algorithm based on 1000 simulation runs and the average of the 4 best results (in terms of objective function) obtained from 1000 budget-constrained random runs.

has experienced a steady, approximately tenfold increase in computational requirements per trained milestone model per year, which has not solely corresponded to advances in hardware capabilities but rather has been mainly associated with an increased push for massive parallelization. Based on these considerations, it is expected that the divide between the capabilities of state-of-the-art cloud/remote edge computing-based AI solutions and those computed directly on mobile devices may continue to increase.

Given the current and expected future trends in the AI accelerator market, it would be useful to introduce a scenario involving accelerators of this kind into the study of MEC hardware resource optimization and compare it with more conventional CPU + GPU hardware combinations.

However, due to the rapid development of the AI accelerator market, such a comparison may be difficult from a cost comparison point of view at the time of writing and is thus left for possible future research.

### C. BLOCKCHAIN ENHANCEMENT BASED ON MEC

Blockchain is one of the key technologies underlying the envisioned future digitized economy. Blockchains will likely support various industries, such as finance, manufacturing, retail, and ICT. MEC servers can act as validators for a blockchain by using their capacity to store the history of transactions [45]. There are multiple applications that can be supported by blockchain technology but have different requirements in terms of speed, security, privacy, scalability, etc. [46]. Future networks should therefore support various blockchain protocols, which will need to work simultaneously while sharing the same computational capacity of MEC servers. Therefore, the enhancement of MEC servers for blockchain, such as through the addition of field programmable gate arrays or application-specific integrated circuits, may be an interesting topic for the future.

### D. 6G EDGE INTELLIGENCE

Whereas MEC currently strongly relies on the powerful 5G infrastructure, we cannot ignore the fact that future generations of mobile networks will require even tighter integration with AI. Edge intelligence empowered by AI is considered one of the key missing elements in current 5G networks and is likely to be considered one of the key enabling factors for future 6G development. 6G network technology is anticipated to encompass hyperdense network deployment, reconfigurable hardware, flying-drone cells, fully programmable network infrastructures, and an immersive user experience [47]. Thus, the importance of MEC and AI will be even greater for the effective management of the growing radio access network (RAN) infrastructure. Therefore, the findings of the current paper are very important for supporting the development of future 6G edge intelligence.

### E. THz OR LI-FI WIRELESS CHANNEL MODELS

In [48], several promising high-frequency bands for wireless communication were listed and compared, among which the THz and Li-Fi bands were both considered potential candidates for use in 6G mobile networks.

Research on the terahertz (THz) band (0.06-10 THz) has been conducted in many studies, and the results have indicated significant improvements in performance [49]. Therefore, the incorporation of THz technology may be a major improvement to the MEC system model, as it can provide higher bandwidths and higher data rates. Additionally, as seen from our simulation results, the wireless network can become a bottleneck as the numbers of users and tasks increase, which will lead to a low QoS for UEs.

In related studies, the THz band has shown high atmospheric absorption; therefore, THz channels are highly frequency selective depending on the designed application [50]. However, the channel model for such a frequency range is still under development. The authors of [51] performed extensive ray-tracing simulations to observe channel characteristics in the range of 275 to 325 GHz. Similarly, in [52] and [53], a D2D scatter channel model was developed, considering statistical geometrical information. In both [54] and [55], a statistical channel model was built in which kiosk application scenarios were specifically considered. The

above studies all focused on close-range communication or indoor communication due to the limited transmission range of THz signals; accordingly, if such channel models are to be implemented in the MEC system model, the range of each AAU will need to be changed, or it may even be necessary to include femtocells or picocells for sufficient coverage.

On the other hand, the Li-Fi band (430-790 THz), which is also referred to as the visible light communication (VLC) or free-space optical (FSO) band, shows even better performance in terms of data rate than the THz band [56]. The characteristics of Li-Fi make it suitable for indoor usage; hence, many researchers have focused on hybrid networks combining Li-Fi and Wi-Fi to achieve sufficient network coverage and significant performance boosts [57]–[59].

## VIII. CONCLUSION

We have studied multitier MEC deployment based on an ultradense 5G network. The main assumptions in the studied scenario are that each active antenna unit is equipped with a low-power MEC server, while each distributed unit is equipped with a high-power MEC server. For the given scenario, we have proposed an algorithm for optimizing the allocation of hardware resources for any given budget constraint. In addition, we have analyzed the performance of the proposed system for various computational requirements and compared it with conventional pseudorandom resource allocation. Simulation results show that the proposed optimization algorithm provides up to 41% higher computational capacity than pseudorandom resource allocation under the same budget. Finally, we have outlined the potential importance of MEC optimization for the future development of AI, blockchain and 6G technologies.

## REFERENCES

[1] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1232–1243, 4th Quart., 2012.

[2] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[3] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[4] L. I. Albraheem, L. H. Alhudaithy, A. A. Aljaser, M. R. Aldhafian, and G. M. Bahliwah, "Toward designing a Li-Fi-Based hierarchical IoT architecture," *IEEE Access*, vol. 6, pp. 40811–40825, 2018.

[5] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multitier fog computing with large-scale IoT data analytics for smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 677–686, Apr. 2018.

[6] (Jan. 2019). *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[7] 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.

[8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[9] X. Ge, S. Tu, G. Mao, and C. X. Wang, "5G ultra-dense cellular networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 72–79, Feb. 2016.

[10] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.

[11] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using M/G/m/m+r queuing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 936–943, May 2012.

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

[13] M. Marjanovic, A. Antonic, and I. P. Zarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. 6, pp. 10662–10674, 2018.

[14] H. Tanaka, M. Yoshida, K. Mori, and N. Takahashi, "Multi-access edge computing: A survey," *J. Inf. Process.*, vol. 26, pp. 87–97, Feb. 2018.

[15] X. Wang, Y. Ji, J. Zhang, L. Bai, and M. Zhang, "Joint optimization of latency and deployment cost over TDM-PON based MEC-enabled cloud radio access networks," *IEEE Access*, vol. 8, pp. 681–696, 2020.

[16] (2014). *International Telecommunication Union Telecommunication Standardization Sector: The Tactile Internet, ITU-T Technology Watch Report*. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/gen/T-GEN-TWATCH-2014-1-PDF-E.pdf

[17] A. Davis, J. Parikh, and W. E. Weihl, "Edgecomputing: Extending enterprise applications to the edge of the Internet," in *Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters*, 2004, pp. 180–187.

[18] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[19] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[20] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2595–2621, 4th Quart., 2018.

[21] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[22] W. Guo. (Oct. 2020). *Nvidia Geforce Now Network Traffic Measurements*. [Online]. Available: https://github.com/the-why-combinator/traffic-data-for-nvidia-now-client

[23] H. Assem Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[24] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.

[25] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 109–116.

[26] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.

[27] X. He, H. Xing, Y. Chen, and A. Nallanathan, "Energy-efficient mobile-edge computation offloading for applications with shared data," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[28] S. Vakilinia, D. Qiu, and M. M. Ali, "Optimal multi-dimensional dynamic resource allocation in mobile cloud computing," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, p. 201, 2014.

[29] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[30] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[31] Y. Sun, D. W. K. Ng, and R. Schober, "Multi-objective optimization for power efficient full-duplex wireless communication systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[32] Y. Sun, D. W. K. Ng, J. Zhu, and R. Schober, "Multi-objective optimization for robust power efficient and secure full-duplex wireless communication systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 8, pp. 5511–5526, Aug. 2016.

[33] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.

[34] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[35] A. ur Rehman Khan, M. Othman, S. Ahmad Madani, and S. Ullah Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.

[36] S. Vakilinia, M. M. Ali, and D. Qiu, "Modeling of the resource allocation in cloud computing centers," *Comput. Netw.*, vol. 91, pp. 453–470, Nov. 2015.

[37] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.

[38] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[39] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Essex, U.K.: Pearson, 2002.

[40] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 115–123.

[41] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller, and M. Upton, "Hyper-threading technology architecture and microarchitecture," *Intel Technol. J.*, vol. 6, no. 1, pp. 4–15, 2002. [Online]. Available: https://services.informatik.hs-mannheim.de/~ihme/lectures/RUR_Files/Hyperthreading.pdf

[42] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[43] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A State-of-the-Art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.

[44] D. Amodei and D. Hernandez. (2018). Ai and Compute. Heruntergeladen von. [Online]. Available: https://blog.openai.com/aiand-compute

[45] R. Sekaran, R. Patan, A. Raveendran, F. Al-Turjman, M. Ramachandran, and L. Mostarda, "Survival study on blockchain based 6G-enabled mobile edge computation for IoT automation," *IEEE Access*, vol. 8, pp. 143453–143463, 2020.

[46] T. Maksymyuk, J. Gazda, M. Volosin, G. Bugar, D. Horvath, M. Klymash, and M. Dohler, "Blockchain-empowered framework for decentralized network management in 6G," *IEEE Commun. Mag.*, vol. 58, no. 9, pp. 86–92, Sep. 2020.

[47] T. Maksymyuk, E. Šlapak, G. Bugár, D. Horváth, and J. Gazda, "Intelligent framework for radio access network design," *Wireless Netw.*, vol. 26, no. 1, pp. 759–774, Jan. 2020.

[48] H. Elayan, O. Amin, R. M. Shubair, and M.-S. Alouini, "Terahertz communication: The opportunities of wireless technology beyond 5G," in *Proc. Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Apr. 2018, pp. 1–5.

[49] I. F. Akyildiz and J. M. Jornet, "Realizing ultra-massive MIMO(1024×1024)communication in the (0.06–10) terahertz band," *Nano Commun. Netw.*, vol. 8, pp. 46–54, Jun. 2016.

[50] J. M. Jornet and I. F. Akyildiz, "Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3211–3221, Oct. 2011.

[51] S. Priebe and T. Kurner, "Stochastic modeling of THz indoor radio channels," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4445–4455, Sep. 2013.

[52] S. Kim and A. Zajic, "Statistical modeling and simulation of short-range device-to-device communication channels at sub-THz frequencies," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6423–6433, Sep. 2016.

[53] S. Kim and A. Zajic, "Statistical modeling of Thz scatter channels," in *Proc. 9th Eur. Conf. Antennas Propag. (EuCAP)*, 2015, pp. 1–5.

[54] D. He, K. Guan, A. Fricke, B. Ai, R. He, Z. Zhong, A. Kasamatsu, I. Hosako, and T. Kurner, "Stochastic channel modeling for kiosk applications in the terahertz band," *IEEE Trans. THz Sci. Technol.*, vol. 7, no. 5, pp. 502–513, Sep. 2017.

[55] A. R. Ekti, A. Boyaci, A. Alparslan, I. Unal, S. Yarkan, A. Gorcin, H. Arslan, and M. Uysal, "Statistical modeling of propagation channels for terahertz band," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Sep. 2017, pp. 275–280.

[56] E. Ciaramella, Y. Arimoto, G. Contestabile, M. Presi, A. D'Errico, V. Guarino, and M. Matsumoto, "1.28 terabit/s (32×40 Gbit/s) wdm transmission system for free space optical communications," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 9, pp. 1639–1645, Dec. 2009.

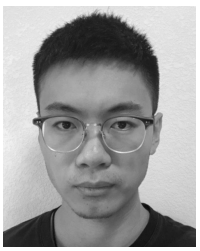
[57] Y. Wang and H. Haas, "Dynamic load balancing with handover in hybrid Li-Fi and Wi-Fi networks," *J. Lightw. Technol.*, vol. 33, no. 22, pp. 4671–4682, Nov. 15, 2015.

[58] X. Wu, M. Safari, and H. Haas, "Access point selection for hybrid Li-Fi and Wi-Fi networks," *IEEE Trans. Commun.*, vol. 65, no. 12, pp. 5375–5385, Dec. 2017.

[59] X. Li, R. Zhang, and L. Hanzo, "Cooperative load balancing in hybrid visible light communications and WiFi," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1319–1329, Apr. 2015.

**EUGEN ŠLAPAK** is currently pursuing the Ph.D. degree with the Technical University of Košice, Slovakia. His Ph.D. thesis focuses on HetNet physical topology design, under the supervision of Assoc. Prof. Juraj Gazda. He works with the Research Team at the university's Intelligent Information Systems Laboratory (http://iislab.kpi.fei.tuke.sk/).

**JURAJ GAZDA** was a Guest Researcher with Ramon Llull University, Barcelona, Spain, and worked under the supervision of Prof. Herman Rohling with the Technical University of Hamburg, Hamburg, Germany. He was also involved in the development activities of Nokia Siemens Networks (NSN). He is currently an Associate Professor with the Faculty of Electrical Engineering, Technical University of Košice, Slovakia. His research interests include spectrum pricing, techno-economic aspects of 5G networks, co-existence of heterogeneous networks, and application of machine learning principles in 5G networks. In 2017, he was recognized as the Best Young Scientist from the Technical University of Košice. He currently serves as an Editor for *KSII Transactions on Internet and Information Systems* and a Guest Editor for *Wireless Communications and Mobile Computing* (Wiley).



**WEIQIANG GUO** (Student Member, IEEE) received the M.S. degree in electronic and electrical engineering from the University of Sheffield, and the M.S. degree in engineering with management from King's College London, in 2016 and 2017, respectively. He is currently pursuing the Ph.D. degree in telecommunications with King's College London under the supervision of Prof. Mischa Dohler. His research interests include mobile edge computing and optimization algorithms.

**TARAS MAKSYMYUK** (Member, IEEE) received the B.A. degree in telecommunications, the M.S. degree in information communication networks, and the Ph.D. degree in telecommunication systems and networks from Lviv Polytechnic National University, Lviv, Ukraine, in 2010, 2011, and 2015, respectively. He is currently an Assistant Professor with the Telecommunications Department, Lviv Polytechnic National University. He did his Postdoctoral Fellowship with the Internet of Things and Cognitive Networks Laboratory, Korea University, under the supervision of Prof. Minho Jo. His research interests include cognitive radio networks, LTE in unlicensed spectrum, mobile cloud computing, massive MIMO, 5G heterogeneous networks, software-defined radio access networks, the Internet of Things, big data, and artificial intelligence. He was recognized as the Best Young Scientist from Lviv Polytechnic National University, in 2015. He received the Lviv State Administration Prize for outstanding scientific achievements and contribution, in 2016, and the Lviv Metropolitan Prize for Best Scientist, in 2017. He is currently an Associate Editor of IEEE *Communications Magazine* and an Editor of the *International Journal of Internet of Things and Big Data*.

**MISCHA DOHLER** (Fellow, IEEE) has worked as a Senior Researcher with Orange/France Telecom, from 2005 to 2008. He was the Director of the Centre for Telecommunications Research, King's College London, from 2014 to 2018. He is currently the Co-Founder of the Smart Cities pioneering company Worldsensing, where he was the CTO, from 2008 to 2014. He is also a Full Professor of Wireless Communications with the King's College London, driving cross-disciplinary research and innovation in technology, sciences, and arts. He is a Serial Entrepreneur, a Composer and a Pianist with five albums on Spotify/iTunes, and fluent in six languages. He acts as a Policy Advisor on issues related to digital, skills, and education. He has more than 300 highly-cited publications and authored several books. He has had ample coverage by national and international press and media. He holds a dozen patents. He is a Fellow of the Royal Academy of Engineering, the Royal Society of Arts (RSA), and the Institution of Engineering and Technology (IET). He is a Distinguished Member of Harvard Square Leaders Excellence. He is a frequent keynote, panel, and tutorial speaker. He has received numerous awards. He has pioneered several research fields, contributed to numerous wireless broadband, the IoT/M2M, and cyber security standards. He has organized and chaired numerous conferences. He was the Editor-in-Chief of two journals.