Collaborative Cloud-Edge-Local Computation Offloading for Multi-Component Applications

Anousheh Gholami
Department of Electrical and Computer Engineering
Institute for Systems Research
University of Maryland, College Park
MD, USA
anousheh@umd.edu

John S. Baras
Department of Electrical and Computer Engineering
Institute for Systems Research
University of Maryland, College Park
MD, USA
baras@umd.edu

ABSTRACT

With the explosion of intelligent and latency-sensitive applications such as AR/VR, remote health and autonomous driving, mobile edge computing (MEC) has emerged as a promising solution to mitigate the high end-to-end latency of mobile cloud computing (MCC). However, the edge servers have significantly less computing capability compared to the resourceful central cloud. Therefore, a collaborative cloud-edge-local offloading scheme is necessary to accommodate both computationally intensive and latency-sensitive mobile applications. The coexistence of central cloud, edge servers and the mobile device (MD), forming a multi-tiered heterogeneous architecture, makes the optimal application deployment very challenging especially for multi-component applications with component dependencies. This paper addresses the problem of energy and latency efficient application offloading in a collaborative cloud-edge-local environment. We formulate a multi-objective mixed integer linear program (MILP) with the goal of minimizing the system-wide energy consumption and application end-to-end latency. An approximation algorithm based on LP relaxation and rounding is proposed to address the time complexity. We demonstrate that our approach outperforms existing strategies in terms of application request acceptance ratio, latency and system energy consumption.

CCS CONCEPTS

• **Networks** → **Network resources allocation**; **Cloud computing**.

KEYWORDS

Mobile edge computing, Collaborative cloud-edge-local computing, Multi-component applications, Integer programming

ACM Reference Format:
Anousheh Gholami and John S. Baras. 2021. Collaborative Cloud-Edge-Local Computation Offloading for Multi-Component Applications. In *The Sixth ACM/IEEE Symposium on Edge Computing (SEC '21), December 14–17, 2021, San Jose, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3453142.3493515

SEC '21, December 14–17, 2021, San Jose, CA, USA

ACM ISBN 978-1-4503-8390-5/21/12…$15.00
https://doi.org/10.1145/3453142.3493515

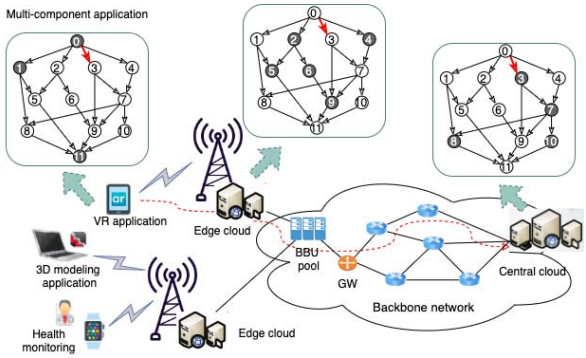

Figure 1: An example of a multi-component application deployment in a collaborative cloud-edge-local environment

1 INTRODUCTION

In recent years, mobile edge computing (MEC) has become a key technology to realize latency-sensitive applications by pushing different resources from remote cloud to network edge in close proximity to mobile users, where data is generated. Prior to MEC, mobile cloud computing (MCC), as an integration of cloud computing and mobile computing, empowered mobile devices with storage, computing, and energy resources provided by the centralized cloud. However, the capability of utilizing a vast amount of idle computation power and storage space distributed at the network edge and the low-latency requirements of modern applications such as autonomous driving, AR/VR, remote health, online gaming, etc., resulted in a shift in computing paradigm, namely MEC [11]. On the other hand, MEC has significantly lower computing and storage capacity compared to the centralized cloud and it easily becomes overloaded. Therefore, it is usually not feasible to offload the computation load of all requested applications to the edge resources and a collaborative cloud-local-edge architecture can potentially enhance users' quality of experience (QoE) [5, 6]. Although the problem of application deployment in a two-tiered computing environment (cloud-local or edge-local) has been extensively explored in the literature, there exist less considerable works on the collaborative cloud-edge-local domain. Moreover, a majority of research have addressed offloading strategies for single tasks in the cloud or edge computing systems [10, 17]. Considering the recent paradigm shift in the application provisioning model from a monolithic service to the microserives architecture, most of the existing solutions need to be re-examined [16] to address the deployment of multi-component

applications which consist of multiple components with arbitrary component dependencies. An instance of such applications and its deployment in a collaborative cloud-edge-local environment is shown in Fig. 1. In this example, components $\{0, 1, 11\}$, $\{2, 4, 5, 6, 9\}$ and $\{3, 7, 8, 10\}$ are executed locally, at edge server, and at the central cloud respectively. In addition to the component assignment, network resources should be provisioned for the transmission of the data between two interacting components in multi-component applications. For instance, network bandwidth across a path from MD to the central cloud is reserved in the example of Fig. 1 to accommodate the data communication between components 0 and 3. Hence a key technical contribution of this paper is proposing a mathematical formulation for multi-component application deployment across a heterogeneous multi-tiered infrastructure.

Furthermore, the energy demand of the communications industry is projected to increase from 200-300 TWh in 2017 to 1200 or even 3000 TWh by 2025 [2] and the global network electricity bill is growing rapidly by 10% each year. Today's network devices are often powered 24/7 for a high availability guarantee of network services. While energy efficiency in cloud data centers and from the MD perspective has been broadly explored, it is largely left uninvestigated in multi-tiered computing environments due to the complicated interactions between MDs, edge servers, and the centralized cloud [7]. In this work, we focus on answering the following questions in a realistic online setting where arrival and service time of applications are not known in advance:

- Which components of an application are computed locally, offloaded to the edge or to the central cloud in order to better utilize available computing, storage and networking resources and achieve higher level of QoS?
- How the above decisions are optimized such that the total compute and network energy consumption is minimized under capacity constraints and MD energy budget?

We design a multi-objective MILP with the objective of minimizing the total consumed energy and end-to-end latency. We also propose an efficient approximation algorithm based on LP relaxation and rounding (LPRR) to solve the problem in polynomial time. Our proposed approach can be used for any multi-component application with arbitrary component dependencies modeled as a directed acyclic graph (DAG).

The paper is organized as follows. We provide an overview of the related works in section 2. Section 3 describes system model, problem formulation and the proposed solution. Performance evaluation is presented in section 4. We highlight our conclusions in section 5.

2 RELATED WORKS

The related research on the MEC computation offloading can be divided into two categories, full offloading and partial offloading. Full offloading has been considered extensively in the literature such as in [10]. Partial offloading which traditionally deals with partitioning the considered task into two subtasks, one running locally and the other one remotely, has also been investigated in the existing works such as [17]. Recently, authors in [3] consider partitioning computation tasks into multiple subtasks each executed locally, at the edge or central cloud and model the latency minimization problem as a MILP with dual decomposition and matching-based algorithms proposed to derive near-optimal solutions. However, the the proposed solution is not applicable to general subtask dependency. In the context of multi-component application which potentially extends partial offloading to arbitrary decomposition of applications, authors in [9] design an integer particle swarm optimization-based algorithm followed by a heuristic for the deployment of code-partitioned and MEC-enabled AR services. In [1], the problem of multi-component application placement in MEC systems is addressed considering the users mobility and network capabilities. The computation offloading problem in a collaborative environment has also been studied in the literature. Authors in [15] study computation offloading in a fog computing network, where the end users offload part of their tasks to a fog node and the fog node further offloads the task to neighboring fog nodes or a remote cloud server. An efficient collaborative task offloading scheme is proposed in [6] in which the MEC server collaborate with MDs and a remote cloud to provide better QoS. In contrast to the existing works targeting single task offloading in collaborative cloud-edge-local domains or multi-component application deployment in edge-local or cloud-local systems, we study the problem of multi-component application deployment in a collaborative multi-tiered environment.

3 SYSTEM MODEL AND PROBLEM FORMULATION

In the sequel we first present the considered system model and then formulate an optimization problem.

3.1 System Model

We assume a collaborative cloud-edge-local environment where the components of a mobile application can be computed locally, in the edge servers or in the central cloud. The proposed scheme is applicable to multi-component applications which according to [20], can be classified into the following three major categories of data-partitioned-oriented, code-partitioned-oriented, and continuous-execution applications. From practical point of view, two settings can be assumed: (i) *offline* scheme in which all application requests are known in advance, and (ii) *online* scheme where the applications arrive and depart the system over time and the requests are processed upon arrival [19]. We consider an online setting which is more realistic and challenging. An edge cloud is defined as a pool of virtualized computing resources, usually co-located with a cellular base station (BS) or a WiFi access point. We consider a cellular system where the edge servers are co-located with BSs and MDs within the coverage of a BS communicate with the corresponding BS (edge server). Moreover, edge servers are connected to a central cloud via multi-hop paths. Let V_N and $V_M = \{0, ..., M\}$ respectively denote the set of backbone network devices and infrastructure compute nodes, where the indices 0, $\{1, ..., M-1\}$ and M stand for the MD, the edge servers, and the central cloud respectively.

3.2 Definitions

Substrate Graph: We model the physical infrastructure as a directed graph $G_S = (V_S, E_S)$ where $V_S = V_M \cup V_N$ and E_S denotes the substrate links. For each server $u \in V_M$, let f_u and W_u^{STO} denote its

computation and residual storage capacity. Moreover, the data rate (in bit/sec) of the link $l \in E_S$ is represented by R_l.

Application Graph: We model the mobile application as a DAG, $G_A = (V_A, E_A)$, where the vertices in V_A denote the application components and an edge $e \in E_A$ represents the data dependency between two components. The required workload and storage of the node $i \in V_A$ and the data size (in bit) requirement of an edge $e \in E_A$ are denoted by $D_i^{CPU}, D_i^{STO}, D_e$, respectively. The application deployment process is considered as mapping the application graph G_A to the substrate graph G_S and it consists of two mappings: (i) *node mapping* which determines the assignment of the application components to substrate nodes, and (ii) *path mapping* that entails the assignment of the application edges to the substrate paths.

3.3 Computation Model

Let t_u^i and E_u^i denote the processing time and energy consumption of component i running on $u \in V_M$. Given the required workload of procedure i, t_u^i is expressed as:

$$t_u^i = \frac{D_i^{CPU}}{f_u}, \quad u \in \{0, 1, ..., M-1\} \quad i \in V_A \tag{1}$$

In addition to the CPU processing time, the queuing delay should also be accounted for the substrate nodes with partially smaller computation capacity. Therefore, a more holistic model addressing the queuing delay is be stated as:

$$t_u^{i,k} = \sum_{j \le k} \frac{D_{j,k}^{CPU}}{f_u}, \quad u \in \{1, ..., M-1\} \quad i \in V_A \tag{2}$$

where k stands for the processing order (based on FCFS) for application graphs components. For a component offloaded to the central cloud, the processing time can be ignored since the computation power of the cloud data center is relatively big compared to the local or edge servers, i.e $f_0 << f_u << f_M, u \in \{1, ..., M-1\}$. Following the model in [13], the energy consumption corresponding to the component i running on the substrate node u is expressed as:

$$E_u^i = \kappa D_i^{CPU} f_u^2 t_u^i \tag{3}$$

where $\kappa f_u^2 t_u^i$ is the energy consumption per CPU cycle and κ is a constant arising from the hardware architecture.

3.4 Communication Model

In this section, we introduce the communication model. Let t_l^e denote the transmission latency corresponding to mapping $e \in E_A$ to the substrate link $l \in E_S$, expressed as:

$$t_l^e = \frac{D_e}{R_l}, \quad \forall e \in E_A, l \in E_S \tag{4}$$

We assume that frequency division duplex (FDD) is used as the transmission mode and W_D and W_U denote the uplink and downlink channel bandwidths respectively. Hence according to the Shannon formula, the achievable data rate of the uplink/downlink (U/D) wireless links can be expressed as:

$$R_l^{D/U} = W_{D/U} log_2(1 + \frac{P_{s(l)}^{TX} d^{-\nu} |h_{D/U}|^2}{N_0}) \tag{5}$$

where $P_{s(l)}^{TX}$ is the transmission power of the transmitter of link l, and the uplink and downlink channels are assumed to be frequency-flat block-fading Rayleigh channels with a block length larger than the maximum latency requirement of the application. Throughout the paper, we refer to the transmitter and receiver of the link l as $s(l), d(l)$. The pathloss between MDs and BSs is modeled as $d^{-\nu}$ where d and ν are the corresponding distance and the pathloss exponent respectively. Furthermore, the uplink and downlink channel fading coefficients are denoted by h_U and h_D modeled as circularly symmetric complex gaussian random variables.

Let $\mathcal{P}_S$ denote the set of K-shortest paths between any pair of nodes $u, v \in V_M, u \neq v$, i.e. $\mathcal{P}_S$ contains all K-shortest paths between MD and edge servers, MD and central cloud, and edge servers and central cloud. We also represent the total energy consumption corresponding to mapping the edge $e \in E_A$ to the substrate path $p_s \in \mathcal{P}_S$ as $E_{p_s}^e$. $E_{p_s}^e$ comprises of the energy consumption for application components processing and data transmission between two components. Assuming that $P_l = P_{s(l)}^{TX} + P_{d(l)}^{RX}$ is the total consumed power of link l's transmitter and receiver, $E_{p_s}^e$ is computed as follows:

$$E_{p_s}^e = \sum_{l \in p_s} P_l t_l^e + \sum_{u \in p_s} P_{on} f(u) \tag{6}$$

where $f(u)$ is an indicator function representing the required power P_{on} to turn on a network node that has been idle. Thus,

$$f(u) = \begin{cases} 0 & u \in V_S \text{ is active} \\ 1 & u \in V_S \text{ is idle} \end{cases} \tag{7}$$

3.5 Optimization Problem

In this section, we present the problem formulation. Given G_A, G_S and $\mathcal{P}_S$, we define the following decision variables for the problem formulation:

- A set of binary decision variables $\boldsymbol{x}$, where x_u^i equals 1 if the application node $i \in V_A$ is mapped to the substrate node $u \in V_M$.
- A set of binary decision variables $\boldsymbol{y}$, where $y_{p_s}^e$ is 1 if the application edge $e \in E_A$ is mapped to the substrate path $p_s \in \mathcal{P}_S$.

The system total consumed energy is computed as:

$$E(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in V_A} \sum_{u \in V_M} E_u^i x_u^i + \sum_{e \in E_A} \sum_{p_s \in \mathcal{P}_S} E_{p_s}^e y_{p_s}^e \tag{8}$$

Let $\mathcal{P}_A$ denote the set of all directed paths in G_A and $p_A \in \mathcal{P}_A$. The overall latency of p_A denoted by $L_{p_A}(\boldsymbol{x}, \boldsymbol{y})$ equals the summation of its nodes (components) processing times and the data transmission delay of its edges, i.e.:

$$L_{p_A}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{e \in p_A} (\sum_{p_s \in \mathcal{P}_S} t_{p_s}^e y_{p_s}^e) + \sum_{i \in p_A} (\sum_{u \in V_M} t_u^i x_u^i) \tag{9}$$

where $t_{p_s}^e = \sum_{l \in p_s} t_l^e + t_l^{prop}$ is the overall latency of mapping the edge $e \in E_A$ to the substrate path $p_s \in \mathcal{P}_S$, and t_l^{prop} is the propagation delay of link $l \in E_S$. Given $\mathcal{P}_A$, we define the *critical path* to be the path inducing the maximum latency among all paths of the application. It is important to note that since the latency incurred by an application component or edge is a function of the available resources of the selected substrate node and path for mapping, it is not trivial to determine the *critical path* in advance.

Therefore, the overall latency of an application (the latency of its *critical path*) is given as:

$$L(\boldsymbol{x}, \boldsymbol{y}) = max_{p_A \in \mathcal{P}_A} L_{p_A}(\boldsymbol{x}, \boldsymbol{y}) \tag{10}$$

The constraints defining the feasible region of our optimization problem are defined below. Starting with the **mapping constraints**, we have:

$$\sum_{u \in V_M} x_u^i = 1, \quad \forall i \in V_A \tag{11}$$

$$x_0^1 = 1, \quad x_0^{|V_A|} = 1 \tag{12}$$

$$\sum_{\substack{v \in V_M \\ p_s(u \to v) \in \mathcal{P}_S}} y_{p_s(u \to v)}^{e(i \to j)} - y_{p_s(v \to u)}^{e(i \to j)} = x_u^i - x_u^j \quad \forall e \in E_A, u \in V_M \tag{13}$$

Constraints (11) ensure that each application node is assigned to one substrate server. In (12), we enforce that the first and last components are computed locally. Constraints (13) guarantee the assignment of paths to the application edges for data communication between interacting components which are mapped to two different substrate nodes. The **MD energy budget constraint** is defined as follows:

$$\sum_{i \in V_A} E_0^i x_0^i + \sum_{v \in V_M} \sum_{e \in E_A} P_0^{TX} t_l^e y_{p_s(0 \to v)}^{e(i \to j)} + \sum_{v \in V_M} \sum_{e \in E_A} P_0^{RX} t_{p_s}^e y_{p(v \to 0)}^{e(j \to i)} \leq RE_0 \tag{14}$$

where RE_0 is the MD's residual energy. The **capacity constrains** are given below:

$$\sum_{i \in V_A} D_i^{STO} x_u^i \leq W_u^{STO}, \quad \forall u \in V_M \tag{15}$$

Finally, the **domain constrains** are given as:

$$x_u^i, y_{p_s}^e \in \{0, 1\}, \quad \forall i \in V_A, e \in E_A, u \in V_M, p_s \in \mathcal{P}_S \tag{16}$$

The objective is to minimize the weighted summation of the total consumed energy and application end-to-end latency. The problem formulation is as follows:

$$[P] \quad \textbf{minimize}_{\boldsymbol{x},\boldsymbol{y}} \quad \lambda \frac{E(\boldsymbol{x}, \boldsymbol{y})}{E_0} + (1 - \lambda) \frac{L(\boldsymbol{x}, \boldsymbol{y})}{L_0} \tag{17}$$
$$\textbf{s.t.} \quad (11) - (16)$$

where E_0 and L_0 are the total consumed energy and application latency when all application components are executed locally and are used to balance the two objective terms. λ is a non-negative constant that determines the tradeoff between energy and latency. In order to linearize (17), we use an auxiliary continuous variable z. It is straightforward to observe that $[P]$ is equivalent to the following MILP:

$$[P'] \quad \textbf{minimize}_{\boldsymbol{x},\boldsymbol{y}} \quad \lambda \frac{E(\boldsymbol{x}, \boldsymbol{y})}{E_0} + (1 - \lambda) \frac{z}{L_0} \tag{18}$$
$$\textbf{s.t.}$$
$$L_{p_A}(\boldsymbol{x}, \boldsymbol{y}) \leq z, \forall p_A \in \mathcal{P}_A \tag{19}$$
$$(11) - (16)$$

Table 1: Default Simulation Parameters

parameter	value
MD uplink/downlink BW	$10MHz$
MD uplink/downlink power	$50, 60dBm$
N_0, ν, κ, K	$-147dBm/Hz$, 2, 10^{-11}, 2
f_0, edge server f_u	1×10^9, $U(5, 10) \times 10^9 cycles/s$
W_0^{STO}, W_u^{STO}	$\mathcal{U}(20, 40)$, $\mathcal{U}(1, 2) \times 10^4$
R_l	$10Mbps$ for wired links

3.6 Proposed Solution

Since the MILP model is known to be NP-hard, we propose an approximation algorithm, namely LPRR, to efficiently solve $[P']$ for large networks, whereby the optimal fractional mapping solutions for $\boldsymbol{x}, \boldsymbol{y}$ are obtained from the LP relaxation. Then we adopt a rounding algorithm proposed in [4], and [18] to find integer solutions of $\boldsymbol{x}$. The integer solutions of $\boldsymbol{y}$ are obtained automatically from the last solved LP.

4 PERFORMANCE EVALUATION

In this section, we first describe the simulation environment and then proceed with evaluation results. We implemented our simulations in Java, using IBM ILOG CPLEX commercial solver to solve the MILP model with branch-and-bound method. Our tests are carried out on a server with an Intel i5 CPU at 2.3 GHz and 8 GB of memory. We evaluate the performance of our collaborative cloud-edge-local (CEL) deployment approach on Digex network topology available at the Topology Zoo [8]. A cloud data center is assumed to be located at San Francisco and edge servers are connected to BSs at Philadelphia, Boston, Miami and Charlotte. We adopt an implementation of Yen's algorithm presented in [14] for K-shortest path generation. 100 MDs are randomly distributed in a $1000m \times 1000m$ region around each BS where the BS is located at the center of the square region. The application requests arrive according to a Poisson process with an average rate of 3 requests per 100 time units. The lifetime of requests has exponential distribution with an average of 1000 time units. Remaining simulation parameters are given in Table I. We run the simulation for 200 applications requested by randomly MDs. The number of application components is uniformly distributed in $[4, 10]$. The dependency type of the application requested by each MD is randomly selected from (i) sequential, (ii) parallel, and (iii) layer-by-layer structures given in [12]. D_i^{CPU} and D_i^{STO} for the first and last components have distributions $\mathcal{U}(0.01, 0.03) \times 10^9$ cycles/s and $\mathcal{U}(1, 10)$ respectively. For the rest of the components, the values for CPU and storage are sampled according to $\mathcal{U}(0.1, 0.5) \times 10^9$ and $\mathcal{U}(10, 30)$. Moreover, $D_e \sim \mathcal{U}(50 - 100)Kb$. The performance of the proposed optimal CEL solution (CEL-CPLEX) and its approximation (CEL-LPRR) is compared with the following schemes and their approximations:

- Edge-local execution (EL): only edge servers are considered for offloading.
- Cloud-local execution (CL): only the central cloud is considered for offloading.

Fig. 2 illustrates the acceptance ratio for CEL-CPLEX, CEL-LPRR, EL-LPRR and CL-LPRR. It is observed that the proposed CEL scheme outperforms the EL and CL strategies significantly, as it admits up

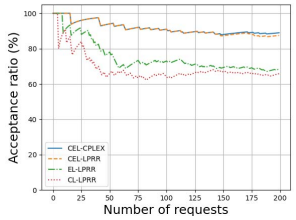

Figure 2: Applications acceptance ratio

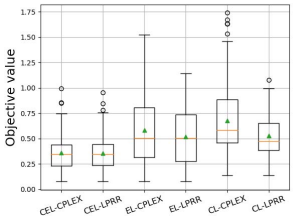

Figure 3: Objective value of $[P']$

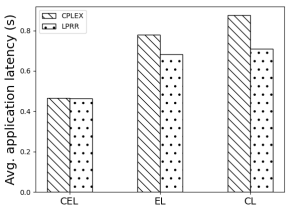

Figure 4: Average application latency

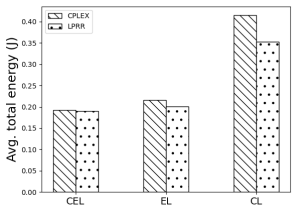

Figure 5: Average total energy

to 23% and 29% more requests than EL-LPRR and CL-LPRR solutions respectively. Moreover, the maximum deviation of CEL-LPRR from CEL-CPLEX is 1.3%. Fig. 3 depicts the box plots corresponding to the objective function values for CEL-CPLEX, EL-CPLEX, CL-CPLEX and their LPRR solutions for the set of accepted requests in each case. It is observed that CEL results in lower objective values compared to EL and CL approaches as expected. Moreover, CEL-LPRR is able to generate near-optimal solutions. It is important to note that CL-LPRR and EL-LPRR schemes have lower average objective values (denoted by green triangles) than CL-CPLEX and EL-CPLEX respectively, since the optimal solutions found by CPLEX solver admits more requests than LPRR algorithm resulting in higher cost (energy and latency) per request. In Fig. 4 and 5, the average overall latency and total energy consumption of CEL is benchmarked against EL and CL solutions. It is observed that the proposed CEL scheme has lower average latency and energy consumption. The four figures together prove the efficiency of the CEL-LPRR approach.

5 CONCLUSION

In this paper, we investigated the optimal computation offloading for multi-component applications in the collaborative cloud-edge-local systems. Our proposed scheme aims at minimizing the total consumed energy and the application end-to-end latency and is applicable to multi-component applications with arbitrary component dependencies. We formulated the problem as a MILP and applied the LP relaxation and rounding technique to generate near-optimal solutions. The simulation results show the superior performance of our proposed solution in terms of acceptance ratio, consumed energy and end-to-end latency compared to two edge-local and central-local offloading baselines.

ACKNOWLEDGMENTS

Research supported in part by ONR grant N00014-17-1-2622 and by a grant from Leidos corporation.

REFERENCES

[1] Tayebeh Bahreini and Daniel Grosu. 2020. Efficient Algorithms for Multi-Component Application Placement in Mobile Edge Computing. *IEEE Transactions on Cloud Computing* (2020), 1–1. https://doi.org/10.1109/TCC.2020.3038626

[2] Sujit Das and Elizabeth Mao. 2020. The global energy footprint of information and communication technology electronics in connected Internet-of-Things devices. *Sustainable Energy, Grids and Networks* (2020), 100408.

[3] Mingjie Feng, Marwan Krunz, and Wenhan Zhang. 2021. Joint Task Partitioning and User Association for Latency Minimization in Mobile Edge Computing Networks. *IEEE Transactions on Vehicular Technology* 70, 8 (2021), 8108–8121. https://doi.org/10.1109/TVT.2021.3091458

[4] Anousheh Gholami, Nariman Torkzaban, John S. Baras, and Chrysa Papagianni. 2021. Joint Mobility-Aware UAV Placement and Routing in Multi-Hop UAV Relaying Systems. In *Ad Hoc Networks*. Springer International Publishing, Cham, 55–69.

[5] Hongzhi Guo, Jiajia Liu, Huiling Qin, and Haibin Zhang. 2017. Collaborative Computation Offloading for Mobile-Edge Computing over Fiber-Wireless Networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 1–6. https://doi.org/10.1109/GLOCOM.2017.8254982

[6] Md Delowar Hossain, Luan NT Huynh, Tangina Sultana, Tri DT Nguyen, Jae Ho Park, Choong Seon Hong, and Eui-Nam Huh. 2020. Collaborative task offloading for overloaded mobile edge computing in small-cell networks. In *2020 International Conference on Information Networking (ICOIN)*. IEEE, 717–722.

[7] Congfeng Jiang, Tiantian Fan, Honghao Gao, Weisong Shi, Liangkai Liu, Christophe Cérin, and Jian Wan. 2020. Energy aware edge computing: A survey. *Computer Communications* 151 (2020), 556–580.

[8] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. 2011. The internet topology zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1765–1775.

[9] Jianhui Liu and Qi Zhang. 2019. Code-Partitioning Offloading Schemes in Mobile Edge Computing for Augmented Reality. *IEEE Access* 7 (2019), 11222–11236. https://doi.org/10.1109/ACCESS.2019.2891113

[10] Xinchen Lyu, Hui Tian, Wei Ni, Yan Zhang, Ping Zhang, and Ren Ping Liu. 2018. Energy-Efficient Admission of Delay-Sensitive Tasks for Mobile Edge Computing. *IEEE Transactions on Communications* 66 (2018), 2603–2616. https://doi.org/10.1109/TCOMM.2018.2799937

[11] Pavel Mach and Zdenek Becvar. 2017. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1628–1656.

[12] S. Eman Mahmoodi, R. N. Uma, and K. P. Subbalakshmi. 2019. Optimal Joint Scheduling and Cloud Offloading for Mobile Applications. *IEEE Transactions on Cloud Computing* 7, 2 (2019), 301–313. https://doi.org/10.1109/TCC.2016.2560808

[13] Yuyi Mao, Jun Zhang, and Khaled B Letaief. 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3590–3605.

[14] Ernesto QV Martins and Marta MB Pascoal. 2003. A new implementation of Yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1, 2 (2003), 121–133.

[15] Mithun Mukherjee, Suman Kumar, Constandinos X. Mavromoustakis, George Mastorakis, Rakesh Matam, Vikas Kumar, and Qi Zhang. 2020. Latency-Driven Parallel Task Data Offloading in Fog Computing Networks for Industrial Applications. *IEEE Transactions on Industrial Informatics* 16 (2020), 6050–6058. https://doi.org/10.1109/TII.2019.2957129

[16] Kaustabha Ray, Ansuman Banerjee, and Nanjangud C. Narendra. 2020. Proactive Microservice Placement and Migration for Mobile Edge Computing. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. 28–41. https://doi.org/10.1109/SEC50012.2020.00010

[17] Min Sheng, Yanting Wang, Xijun Wang, and Jiandong Li. 2020. Energy-Efficient Multiuser Partial Computation Offloading With Collaboration of Terminals, Radio Access Network, and Edge Server. *IEEE Transactions on Communications* 68, 3 (2020), 1524–1537. https://doi.org/10.1109/TCOMM.2019.2959338

[18] Nariman Torkzaban, Anousheh Gholami, John S. Baras, and Chrysa Papagianni. 2020. Joint Satellite Gateway Placement and Routing for Integrated Satellite-Terrestrial Networks. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC40277.2020.9149175

[19] Feng Wang, Jie Xu, Xin Wang, and Shuguang Cui. 2017. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications* 17, 3 (2017), 1784–1797.

[20] Yanting Wang, Min Sheng, Xijun Wang, Liang Wang, and Jiandong Li. 2016. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications* 64, 10 (2016), 4268–4282.