Energy-Efficient Resource Allocation for D2D-Assisted Fog Computing

Onur Karatalay, Ioannis Psaromiligkos and Benoit Champagne
Department of Electrical and Computer Engineering, McGill University, Montréal, QC, Canada.
Email: onur.karatalay@mail.mcgill.ca; ioannis.psaromiligkos@mcgill.ca; benoit.champagne@mcgill.ca

***Abstract*—In this paper, we address the problem of energy-efficient resource allocation in a multi-device D2D-assisted fog computing scenario, where the goal is to minimize the total energy consumption subject to constraints on the transmit powers, computation resources and task processing times. The considered problem is non-convex and finding its global optimum is generally intractable; hence we propose two sub-optimal approaches to solve it. First, by investigating the relationship between the task processing time and the total energy consumption, we show how the original problem can be relaxed into a sequence of convex subproblems whose solutions can be efficiently obtained via standard algorithms. Second, to further reduce computational complexity, we propose a low-complexity heuristic resource allocation strategy which does not require calculating gradients and the Hessian matrices in the solution process. We also develop a lower bound on the total energy consumption for the considered task offloading scenario as a benchmark for comparison purpose. Computer simulations under a wide range of conditions and parameter settings show that both methods achieve a near-optimal solution in comparison to the lower bound.**

***Index Terms*—D2D, resource allocation, fog computing, task offloading, 5G**

I. Introduction

By the end of 2023, there should be more than 13 billion mobile wireless devices worldwide, which represents a nearly 50% increase since 2018 [1]. Along with the proliferation of connected devices, the anticipated increase in computation-intensive and latency-sensitive mobile applications such as online gaming and virtual reality, will challenge the computation capabilities of mobile devices [2–4] due to the rigid processing deadlines imposed by these applications. Recently, *task offloading*, which allows devices to utilize more powerful remote computation resources for task processing, has attracted considerable interest [5], [6].

A. Related Works and Motivation

Cloud computing, one of the most studied task offloading frameworks, allows mobile device data to be transmitted and processed at remote servers through the internet [7], [8]. However, unpredictable wireless channel conditions and high data traffic due to excessive user density limit the overall task offloading speed, which in turn, reduces the Quality of Service (QoS) [9]. Mobile Edge Computing (MEC) as a substitute to cloud computing, brings data processing closer to the mobile devices by locating data storage and computation servers at the edge of the wireless network. These so-called Edge Servers (ES)s are usually connected to Base Stations (BS)s via high-speed links [7], [8], [10]. Overall, MEC can lower end-to-end delays, reduce data processing bottlenecks and increase energy efficiency within the network. In [11], the authors consider joint task offloading and resource allocation in a MEC scenario, in which the available resources are allocated to minimize the total overhead in terms of both energy consumption and task processing time. The authors in [12] focus on total energy minimization subject to constraints on the task processing times in an ultra-dense network. In [13], the authors propose a resource allocation method for MEC aiming to minimize the maximum task processing delays among all the users, while in [14], energy consumption, task execution delay and cost of task offloading are jointly optimized by using queuing theory. In [15], energy-efficient resource allocation is investigated for latency-sensitive tasks in a multi-user offloading scenario. In [16], the authors study a similar scenario, in which offloading decisions are modeled by using a game-theoretic approach. Moreover in [17], a distributed power allocation method is proposed based on a game theoretic approach, and similarly in [18], minimization of energy consumption and average response time is investigated using game theory. While shedding light on the energy efficiency of utilisation MEC, the aforementioned studies only consider a single off-loading destination, such as a single ES. However, the limited computational capability of ESs (compared to cloud servers) limit the energy efficiency and restrains the performance of task off-loading as data communication and processing demands increase within the network.

As a complement to MEC and cloud computing, fog computing can exploit the full potential of distributed data processing by expanding the pool of computation resources to include nearby mobile devices in addition to Edge/Cloud servers and by allowing data-generating devices to offload their tasks to them through incentive policies [19–22]. This strategy, which leverages the availability of all computation resources within radio proximity, referred to as "fog devices", can significantly decrease the total energy consumption and task processing time. Device-to-Device (D2D) communications, which are envisaged to play a key role in the fifth (5G) and future generations of wireless networks, provide an attractive technology enabler for fog computing, by facilitating the creation of direct communication links between neighboring devices. In [23], the authors maximize the number of devices

This work was supported in part by the Natural Science and Engineering Research Council of Canada under the Discovery Grant program.

in a D2D-aided cellular network subject to constraints on both communication and computation resources. Besides, in [24], the authors study computation latency minimization in the case of a D2D-enabled MEC system, while in [25] the authors focus on time-average energy minimization. Nonetheless, due to the difficulties posed by the optimal allocation of communication and computation resources as the number of offloading destination increases, the above studies only consider a single D2D connection per device within a MEC system. However, restricting the number of D2D connections limits the potential gains in computation capacity and energy efficiency of task offloading. Therefore, a scalable approach, in which an arbitrary number of fog devices can be accessed by multiple D2D connections in a MEC scenario is needed.

Another factor that directly affects the energy consumption is whether the task offloading scheme is binary or partial. In the former case, a task is either computed locally (i.e., on the data generating device) or offloaded entirely to a single neighboring device. In the latter case, a task is divided into various portions for parallel computing on different devices, including the local one. In [26], ES selection in binary task offloading is tackled by minimizing a network access-based cost function under delay and QoS constraints. In [27], the authors consider binary task offloading in a D2D-assisted fog computing scenario, and propose new algorithms based on branch-and-price for jointly optimizing link scheduling, channel assignment and power control. In [25], the authors propose an optimization framework for binary offloading in order to minimize the time-average energy consumption for execution of all user tasks while taking into account fair computation resource allocation as constraints. Likewise in [28], the overall system utility is maximized with respect to binary offloading decisions by developing a pricing game-based algorithm.

In contrast to the binary scheme, partial task offloading can take advantage of parallel computation to increase energy efficiency [15]. In [29], partial task offloading is considered in MEC, where the aim is to minimize the total energy consumption by jointly optimizing the transmit power, computation speed and task partitioning. In [30], the authors investigate a cooperative partial task offloading scheme with both cloud computing and MEC, wherein task partitioning decisions are made to minimize the end-to-end delay. While offering valuable insights into partial task offloading, these works do not take advantage of D2D-aided fog computing to further improve energy efficiency and reduce delay. In contrast, references [31–33] explicitly focus on partial task offloading with the help of D2D communications. In [31], a hybrid D2D-aided fog and cloud computing scenario with a single task offloading device is investigated, whereas [32], [33] focus on a more general multi-device D2D aided fog computing scenario. In these works, the task offloading decisions are made by minimizing the total energy consumption subject to relevant constraints, especially the task processing time. Although significant improvements in energy efficiency can be achieved with D2D-aided fog computing, these works do not consider the allocation of transmit powers and utilization of an ES, which could further enhance the performance of the task offloading.

B. Main Contributions

In this work, motivated by the aforementioned studies, we focus on a more general fog computing scenario, wherein multiple devices can offload their tasks to nearby fog devices via D2D links *and* to an ES. We adopt the partial task offloading scheme with transmit power management to improve utilization of resources, especially, to minimize the total energy consumption over the considered network. On the one hand, the ES has more computation capability compared to the fog devices, and hence can process more tasks simultaneously, although achievable data rates on the ES links may limit the task uploading speed. On the other hand, D2D-aided fog computing can take advantage of close proximity, and hence yields higher data rates and reduces the task offloading time. Within this extended framework, the distinctive contributions of this paper are summarized as follows:

- We consider a D2D-assisted fog computing scenario, where multiple cellular devices can partially offload their computation-intensive tasks not only to the ES but also to nearby fog devices via D2D links. Our main objective is to develop an optimal resource allocation strategy by minimizing the total energy consumption subject to constraints on transmit powers, computation resources and task processing times.
- The formulated problem is non-convex and its optimal solution is generally intractable; in response, we propose two sub-optimal methods. For the first method, we begin with investigating the relationship between the task processing time and the total energy consumption. By exploiting this relationship, we then show how the original problem can be relaxed into a sequence of convex subproblems whose solutions can be efficiently obtained via standard convex optimization methods.
- While our first method achieves good performance, its run time may be high. To remedy this, we propose a second method, which targets similar goals as the first one, but relies on a low-complexity heuristic resource allocation strategy, thereby avoiding costly calculations of gradients and Hessian matrices in the solution process. We analyze in detail the computational complexity of this method in terms of key system parameters, including the number of mobile devices and task offloading destinations.
- We develop a lower bound on the total energy consumption for the considered task offloading scenario as a performance benchmark for comparison purpose. Computer simulations under a wide range of conditions and parameter settings show that both methods approach the lower bound for a wide range of practical conditions, while the second method leads to a quite significant reduction in run time.

The rest of the paper is organized as follows. In Section II, we present the system model and formulate the problem statement. In Section III, we analyze the original problem and develop our first method for task offloading based on

convex programming. In Section IV, we derive our second method, which relies on a low-complexity heuristic resource allocation algorithm. Section V presents the simulation results and accompanying discussions, while Section VI summarizes our findings.

Notation: Boldface letters, e.g., $\mathbf{a}$, are used to denote column vectors, $[\cdot]^\top$ represents the matrix transpose operation, and $\|\cdot\|_p$ stands for the ℓ_p norm. Sets are designated by calligraphy letters, e.g., $\mathcal{N}$, with their cardinality denoted by $|\mathcal{N}|$.

II. System Model and Problem Statement

In this section, we first present the D2D-aided fog computing scenario under study and the associate computation and communication models. We then formulate the problem of optimal resource allocation as a non-convex program.

A. System Model

We consider a stationary network consisting of a single BS and I active cellular devices with computationally intensive tasks. Each active device (indexed by i) can offload its task to a single ES (indexed by 0) connected to the BS via a high-speed link, and up to K nearby fog devices via D2D links, as shown in Fig. 1. We assume that all active devices can simultaneously utilize the ES, however the fog devices are not shared among the active devices. We identify the available task offloading destinations of the ith active device by a 2-tuple $\kappa \in \mathcal{K}_i = \{(i,0),\ (i,1),...,(i,K)\},\ i \in \mathcal{I} = \{1,...,I\}$. In the sequel, to simplify notations, we represent the 2-tuple (i,j) simply as ij. Similar to [34–36], we assume interference-free links among D2D devices as well as between the active devices and the BS, which can be achieved, for example, by allocating orthogonal communication resources to the active devices with the help of the BS. In addition, the assignment of fog devices to the active ones is decided beforehand based on various criteria such as distance, availability, incentive, etc.

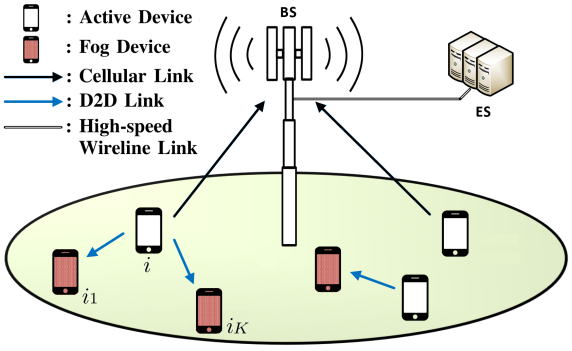


Fig. 1. D2D-aided fog computing scenario, where active devices can offload their tasks to nearby fog devices as well as to a central ES.

The computation task of the ith active device is characterized by the tuple $(d_i, c_i, t_i^{\max})$. Here, d_i indicates the task size in bits, c_i denotes the average number of CPU cycles required to process one bit of data, and $t_i^{\max}$ is the task processing deadline[1] for the task. An active device may offload parts of its task to available destinations. Accordingly, the task size of the ith device can be decomposed as:

$$d_i = b_i + \sum_{\kappa \in \mathcal{K}_i} b_\kappa \tag{1}$$

where b_i and b_κ indicate the sizes of the task portions kept at the local device and sent to the κth offloading destination, respectively. The task portion sizes for the ith active device form the vector $\boldsymbol{b}_i = [b_i\ b_{i0}\ b_{i1}\ ...\ b_{iK}]^\top$ with $\|\boldsymbol{b}_i\|_1 = d_i$.

Let f_i, expressed in cycles per second, represent the computation resources allocated by the ith active device to process the local portion of its task. Then, we can calculate the time taken to compute the local portion of the task as follows:

$$t_i^{\text{co}} = \frac{b_i c_i}{f_i},\ \forall i \in \mathcal{I}. \tag{2}$$

Accordingly, the energy consumed for processing this task is computed as:

$$E_i^{\text{loc}} = \mu b_i c_i f_i^2,\ \forall i \in \mathcal{I} \tag{3}$$

where μ is an effective capacitance constant depending on the chip architecture of the devices [17].

During task offloading, the ith active device transmits b_κ bits to its κth offloading destination over a wireless link at a data rate given by:

$$R_\kappa = W \log_2\left(1 + \frac{P_\kappa G_\kappa}{N_0}\right),\ \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{4}$$

where P_κ is the allocated transmit power, G_κ is the channel gain, W is the channel bandwidth, and N_0 is the thermal noise power. Then we calculate the time needed to complete the transmission of the offloaded task size portion as follows:

$$t_\kappa^{\text{up}} = \frac{b_\kappa}{R_\kappa},\ \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{5}$$

Let $f_\kappa \le f_\kappa^{\max}$ be the computation resources allocated by the κth destination to process the task assigned to it, which cannot exceed its maximum computation capability $f_\kappa^{\max}$, $\forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I}$. Similar to (2), the time taken to compute the offloaded portion of the task is defined as follows:

$$t_\kappa^{\text{co}} = \frac{b_\kappa c_i}{f_\kappa},\ \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{6}$$

and the energy consumed for uploading and processing the task at the offloading destinations of the ith device is given by:

$$E_i^{\text{off}} = \sum_{\kappa \in \mathcal{K}_i} \left(P_\kappa t_\kappa^{\text{up}} + \mu b_\kappa c_i f_\kappa^2\right),\ \forall i \in \mathcal{I} \tag{7}$$

Finally, the total energy consumption required to complete the overall task for the ith active device is computed as:

$$E_i = E_i^{\text{loc}} + E_i^{\text{off}},\ \forall i \in \mathcal{I} \tag{8}$$

[1]That is, the latest time by which processing, including transmission time if applicable, must be completed.

B. Problem Statement

In this paper, we focus on a resource allocation problem for the D2D-aided fog computing scenario described earlier. Specifically, we aim at identifying the optimal task offloading strategy that minimizes the total energy consumption subject to limits on computation resources, transmit powers and task processing times. The overall problem is formally described as follows:

$$\mathscr{P}_1 : \min_{\mathbf{p},\mathbf{f},\mathbf{b}} \quad \sum_{i\in\mathcal{I}} E_i \tag{9a}$$

$$\text{s.t.} \quad 0 \le \sum_{\kappa\in\mathcal{K}_i} \|b_\kappa\|_0 P_\kappa \le P^{\max},\ \forall i \in \mathcal{I} \tag{9b}$$

$$\sum_{i\in\mathcal{I}} \|b_{i0}\|_0 f_{i0} \le f_0^{\max}, \tag{9c}$$

$$f_\kappa \le f_\kappa^{\max}, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i - \{i0\} \tag{9d}$$

$$b_i + \sum_{\kappa\in\mathcal{K}_i} \|P_\kappa\|_0 b_\kappa = d_i,\ \forall i \in \mathcal{I} \tag{9e}$$

$$0 \le \frac{b_i c_i}{f_i} \le t_i^{\max},\ \forall i \in \mathcal{I} \tag{9f}$$

$$0 \le \frac{b_\kappa}{R_\kappa} + \frac{b_\kappa c_i}{f_\kappa} \le t_i^{\max}, \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{9g}$$

$$0 \le b_i, b_\kappa, f_i, f_\kappa, P_\kappa \quad \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{9h}$$

where the vector $\mathbf{p} = [\mathbf{p}_1^\top\ \mathbf{p}_2^\top\ ...\ \mathbf{p}_I^\top]^\top$ with $\mathbf{p}_i = [P_{i0}\ P_{i1}\ ...\ P_{iK}]^\top$ contains the allocated transmit powers of each active device to its offloading destinations. In addition, the vector $\mathbf{f} = [\mathbf{f}_1^\top\ \mathbf{f}_2^\top\ ...\ \mathbf{f}_I^\top]^\top$ with $\mathbf{f}_i = [f_i\ f_{i0}\ f_{i1}\ ...\ f_{iK}]^\top$ contains the allocated computation resources, and the matrix $\mathbf{b} = [\boldsymbol{b}_1^\top\ \boldsymbol{b}_2^\top\ ...\ \boldsymbol{b}_I^\top]^\top$ contains the task splitting decisions of the active devices.

In problem $\mathscr{P}_1$, constraint (9b) limits the total transmit power at the ith device to $P^{\max}$, while constraints (9c) and (9d) restrict the ES and fog devices to allocate computation resource beyond their maximum computation capability $f_0^{\max}$ and $f_\kappa^{\max}, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i - \{i0\}$, respectively. Constraint (9e) ensures that the task splitting adds up to original task size, while constraints (9f) and (9g) require that the time to complete the task does not exceed the task processing deadline $t_i^{\max}\ \forall i \in \mathcal{I}$. Finally, constraint (9h) denotes that the decision variables must be non-negative.

At this point, problem $\mathscr{P}_1$ can be re-formulated as a mixed-integer program to avoid ℓ_0 norms, however, non-convex objective functions and constraints still impose difficulty as they make the problem intractable. Although various techniques are available for solving the non-convex problem $\mathscr{P}_1$, such as particle swarm optimization, genetic algorithm, and exhaustive search, these techniques exhibit very slow convergence due to the large search space, as we have been able to verify. Therefore, in the next section, we propose a sub-optimal method, which depends solely on convex programming, to efficiently solve problem $\mathscr{P}_1$.

III. Convex Programming Method

In this section, we take a closer look at problem $\mathscr{P}_1$ and show how it can be relaxed into convex sub-problems.

A. Problem Analysis

Lemma 1: At the optimal solution of problem $\mathscr{P}_1$ the task processing times must be equal to the deadline.

Proof: Let $\mathbf{p}^\star, \mathbf{f}^\star$ and $\mathbf{b}^\star$ be the global minimizers of problem $\mathscr{P}_1$ yielding a total energy consumption $E^\star = E^{\text{loc}^\star} + E^{\text{off}^\star}$. At the optimal solution, we have $d_i = b_i^\star + \sum_{\kappa\in\mathcal{K}_i} b_\kappa^\star$ and the computation resources corresponding to these task portions are $f_i^\star$ and $f_\kappa^\star$, respectively. Assume that the optimal task processing time for the local portion of the task is smaller than the task processing deadline, i.e., $t_i^{\text{co}^\star} = \frac{b_i^\star c_i}{f_i^\star} < t_i^{\max}$. Then there is a $f_i^\ddagger < f_i^\star$ such that $t_i^{\text{co}^\star} < \frac{b_i^\star c_i}{f_i^\ddagger} = t_i^{\max}$, which yields $E_i^{\text{loc}^\ddagger} = \mu b_i^\star c_i {f_i^\ddagger}^2 < E_i^{\text{loc}^\star}$, and consequently reduces the total energy consumption $E_i^\ddagger = E^{\text{loc}^\ddagger} + E^{\text{off}^\star} < E_i^\star$ further. Therefore, $f_i^\star$ cannot be the global minimizer. Similarly, assume that the optimal total task offloading time is smaller than the task processing deadline, $t_\kappa^{\text{up}^\star} + t_\kappa^{\text{co}^\star} = t_\kappa^{\text{off}^\star} < t_i^{\max}$, or equivalently by using (5) and (6), we have $\frac{b_\kappa^\star}{R_\kappa^\star} + \frac{b_\kappa^\star c_i}{f_\kappa^\star} = t_\kappa^{\text{off}^\star} < t_i^{\max}$, where $R_\kappa^\star$ is the data rate calculated by using the optimal transmit power $P_\kappa^\star$. However, there is a $f_\kappa^\ddagger < f_\kappa^\star$ such that $\frac{b_\kappa^\star}{R_\kappa^\star} + \frac{b_\kappa^\star c_i}{f_\kappa^\ddagger} = t_i^{\max}$ or $P_\kappa^\ddagger < P_\kappa^\star$ such that $\frac{b_\kappa^\star}{R_\kappa^\ddagger} + \frac{b_\kappa^\star c_i}{f_\kappa^\star} = t_i^{\max}$, in which both $f_\kappa^\ddagger$ and $P_\kappa^\ddagger$ yield smaller energy consumption for task offloading, i.e., $E_i^{\text{off}^\ddagger} < E_i^{\text{off}^\star}$. Consequently, $f_\kappa^\star$ and $P_\kappa^\star$ cannot be the optimal minimizers since the total energy consumption in (8) can be reduced further. □

Based on the above, we can modify the objective function (9a) by replacing t_i^{co} with $t_i^{\max}, \forall i \in \mathcal{I}$ and t_κ^{co} with $(t_i^{\max} - t_\kappa^{\text{up}}), \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$, and write the total energy consumption of the ith device as a combination of two terms $E_i = \xi_i(\mathbf{p}_i, \mathbf{b}_i) + \psi_i(\mathbf{p}_i, \mathbf{b}_i), i \in \mathcal{I}$. Specifically, the first term is the total energy consumption for uploading the task size portions:

$$\xi_i(\mathbf{p}_i, \mathbf{b}_i) = \sum_{\kappa\in\mathcal{K}_i} \frac{P_\kappa b_\kappa}{R_\kappa} \tag{10}$$

while the second term is the total computation energy and obtained from (3) and (7) as follows:

$$\psi_i(\mathbf{p}_i, \mathbf{b}_i) = \frac{\mu(b_i c_i)^3}{(t_i^{\max})^2} + \sum_{\kappa\in\mathcal{K}_i} \frac{\mu(b_\kappa c_i)^3}{\left(t_i^{\max} - \frac{b_\kappa}{R_\kappa}\right)^2} \tag{11}$$

Note that reducing both terms simultaneously in E_i could not be possible since decreasing transmit power P_κ might reduce the first term, but it increases the second one due to elevated uploading time $t_\kappa^{\text{up}} = \frac{b_\kappa}{R_\kappa}$. Fortunately, *the total computation energy* given in (11) is a convex function on a convex domain as demonstrated in Appendix A. Next, by leveraging this convexity, we will decompose problem $\mathscr{P}_1$ into convex sub-problems that can be solved using standard techniques.

B. Allocation of Transmit Powers and Task Sizes

First, we allocate the transmit powers and task sizes. Specifically, we determine the optimal task splitting under transmit power constraints that, in turn, limit the data rates, without

considering the available computation resources at the ES and fog devices:

$$\mathscr{P}_2 : \min_{\mathbf{p},\mathbf{b}} \sum_{i\in\mathcal{I}} \psi_i(\mathbf{p}_i, \mathbf{b}_i) \tag{12a}$$

$$\text{s.t.} \quad 0 \leq \sum_{\kappa\in\mathcal{K}_i} P_\kappa \leq P^{\max},\ \forall i \in \mathcal{I} \tag{12b}$$

$$b_i + \sum_{\kappa\in\mathcal{K}_i} b_\kappa = d_i,\ \forall i \in \mathcal{I} \tag{12c}$$

$$b_\kappa - \alpha R_\kappa t_i^{\max} \leq 0,\ \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i \tag{12d}$$

$$0 \leq b_i, b_\kappa \quad \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{12e}$$

In constraint (12d) we include a factor $\alpha \in (0,1)$ to limit the task uploading time and provide sufficient time for task computation at the offloading destinations $t_\kappa^{\text{co}} = (1-\alpha)t_i^{\max}$. This naturally prevents the violation of constraint (9g) in problem $\mathscr{P}_1$, and consequently, allows the solution of problem $\mathscr{P}_2$ to be in the feasible solution set of the main problem. However, we note that the choice of α cannot be arbitrary since it plays an important role in the convexity of problem $\mathscr{P}_2$ as demonstrated in Appendix A. Then, we allocate the computation resources corresponding to the transmit powers and the task splitting decisions $\mathbf{p}^*$ and $\mathbf{b}^*$ obtained by solving problem $\mathscr{P}_2$.

C. Allocation of Computation Resources

To determine the computation resources corresponding to $\mathbf{p}^*$ and $\mathbf{b}^*$, we first calculate the task uploading times using (5) as $t_\kappa^{\text{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}, \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i$, where R_κ^* is the data rate calculated by using the transmit power P_κ^* (cf. eq. (4)). Then, using the remaining time $t_\kappa^{\text{co}^*} = t_i^{\max} - t_\kappa^{\text{up}^*}$, we allocate the computation resources at the κth offloading destination as follows:

$$f_\kappa^* = \frac{b_\kappa^* c_i}{t_i^{\max} - t_\kappa^{\text{up}^*}},\ \forall i \in \mathcal{I}\ \forall \kappa \in \mathcal{K}_i. \tag{13}$$

Similarly, the allocated computation resource at the ith device is:

$$f_i^* = \frac{b_i^* c_i}{t_i^{\max}},\ \forall i \in \mathcal{I}. \tag{14}$$

Using (13) and (14) we form the optimal allocated computation resource vector $\mathbf{f}^*$ in the same manner as $\mathbf{f}$. Recall that problem $\mathscr{P}_2$ does not consider constraints (9c) and (9d), therefore, it may happen that the allocated task sizes are such that $\sum_{i\in\mathcal{I}} \frac{b_{i0}^* c_i}{t_i^{\max} - \frac{b_{i0}^*}{R_{i0}^*}} > f_0^{\max}$ or $\frac{b_\kappa^* c_i}{t_i^{\max} - \frac{b_\kappa^*}{R_\kappa^*}} > f_\kappa^{\max}$, for some $i \in \mathcal{I}, \kappa \in \mathcal{K}_i - \{i0\}$. Consequently, constraints (9c) or (9d) are violated. In that case, we reallocate the task sizes $\mathbf{b}^*$ subject to constraints (9c) and (9d) by fixing the transmit powers to $\mathbf{p}^*$ as follows:

$$\mathscr{P}_3 : \min_{\mathbf{b}} \sum_{i\in\mathcal{I}} \psi_i(\mathbf{p}_i^*, \mathbf{b}_i) \tag{15a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}} \frac{b_{i0} R_{i0}^* c_i}{R_{i0}^* t_i^{\max} - b_{i0}} \leq f_0^{\max} \tag{15b}$$

$$\frac{b_\kappa R_\kappa^* c_i}{R_\kappa^* t_i^{\max} - b_\kappa} \leq f_\kappa^{\max}, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i - \{i0\} \tag{15c}$$

$$b_i + \sum_{\kappa\in\mathcal{K}_i} b_\kappa = d_i,\ \forall i \in \mathcal{I} \tag{15d}$$

$$b_\kappa - \alpha t_i^{\max} R_\kappa^* \leq 0, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{15e}$$

$$0 \leq b_i, b_\kappa, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{15f}$$

We denote the new task size allocations obtained by solving problem $\mathscr{P}_3$ as $\mathbf{b}^+$. Then, we update the task uploading times as $t_\kappa^{\text{up}^+} = \frac{b_\kappa^+}{R_\kappa^*}, \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i$. Finally, we calculate the new allocated computation resources $\mathbf{f}^+$ as in (13) and (14) by using $\mathbf{b}^+$ and $\mathbf{p}^*$.

D. Summary of the Proposed Method

In Algorithm 1 we summarize the proposed convex-programming-based sub-optimal method to solve problem $\mathscr{P}_1$. The algorithm runs in a central location, for example at a BS, where it first obtains the initial task sizes $\mathbf{b}^*$ and the transmit power levels $\mathbf{p}^*$ by solving problem $\mathscr{P}_2$ in step 1, and then, it calculates the corresponding computation resources $\mathbf{f}^*$ in step 3. If constraints (9c) or (9d) are violated, the BS solves problem $\mathscr{P}_3$ in step 5 to update $\mathbf{f}^*$ as $\mathbf{f}^+$ based on $\mathbf{b}^+$. Finally, the obtained solution is relayed to the active devices.

Algorithm 1 Convex Programming Method

1: Solve Problem $\mathscr{P}_2$ to obtain initial $\mathbf{b}^*$ and $\mathbf{p}^*$
2: Calculate $t_\kappa^{\text{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}\ \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$
3: Calculate $\mathbf{f}^*$ based on $\mathbf{b}^*$ and $\mathbf{p}^*$ by using (13) and (14)
4: **if** Constraint (9c) or (9d) are violated **then**
5: Update the task sizes by solving Problem $\mathscr{P}_3$
6: Calculate $t_\kappa^{\text{up}^+} = \frac{b_\kappa^+}{R_\kappa^*}\ \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$
7: Calculate $\mathbf{f}^+$ based on $\mathbf{b}^+$ and $\mathbf{p}^*$ by using (13) and (14)
8: **end**

IV. Heuristic Task Offloading Method

In this section, we propose a heuristic algorithm to solve problem $\mathscr{P}_1$. The main goal is to provide an accurate, low-complexity method that does not require the computation of gradients and Hessian matrices in the solution process. To develop the heuristic method we will follow a sequential approach as in Section III. We start by allocating the transmit powers and the task sizes under data rate constraints, as we did in problem $\mathscr{P}_2$. Then, we take the maximum computation capability of the ES and the fog devices into account as in problem $\mathscr{P}_3$ to prevent the violation of constraints (9c) and (9d).

A. Initial Resource Allocation

Lemma 2: In an ideal scenario where the task uploading is instantaneous and the offloading destinations have infinite computation resources, a lower bound on the total energy consumption is achieved when the task sizes are equally divided among the offloading destinations.

Proof: Please refer to Appendix B.

Based on this, we first initialize the task sizes for the ith active device as follows:

$$\mathbf{b}_i^* = d_i(|\mathcal{K}_i|+1)^{-1}\mathbf{1},\ \forall i \in \mathcal{I} \tag{16}$$

where $\mathbf{1}$ is the column vector of all-ones. If task uploading from each active device to its offloading destinations were instantaneous, then the total energy consumption based on the initial task splitting strategy in (16) would clearly attain the lower bound derived in Appendix B. However, in a realistic scenario, due to wireless channel conditions as well as the transmit power constraints, the achievable data rates are limited. Hence, our next step is to reduce the task uploading time as much as possible to approach the lower bound. To this end, we allocate the transmit powers to compensate for the channel conditions between the ith active device and its offloading destinations:

$$\mathbf{p}_i^* = \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|_1} P^{\max},\ \forall i \in \mathcal{I} \tag{17}$$

where $\mathbf{h}_i = \|\mathbf{g}_i\|_1\mathbf{1} - \mathbf{g}_i$ is the vector that contains the channel gains of the ith device that are subtracted from its ℓ_1-norm and the channel gain vector is defined as $\mathbf{g}_i = \left[G_{i0}\ G_{i1}\ ...\ G_{iK}\right]^\top,\ i \in \mathcal{I}$. By initializing the transmit powers as in (17), each active device allocates more transmit power to its offloading destinations with relatively low channel gain to increase data rates, and in turn, reduce task uploading time. However, with the initial task sizes and transmit power allocation, the uploading time from ith device to its κth offloading destination might exceed the task processing deadline, i.e., $t_\kappa^{\text{up}^*} > t_i^{\max}$, which violates constraint (9g).

Since $\mathbf{p}^*$ is initialized to compensate for the physical channel conditions, we update the initial task partitioning $\mathbf{b}^*$ to have a feasible $t_\kappa^{\text{up}^*}$ similar to constraint (12d). Specifically, for any active device i and its κth offloading destination for which $t_\kappa^{\text{up}^*} > \alpha t_i^{\max}$, we set $t_\kappa^{\text{up}^*} = \alpha t_i^{\max}$, which is the maximum time limit for uploading the task sizes as in constraint (12d). Then, we re-calculate the corresponding task size as $b_\kappa^+ = R_\kappa^* \alpha t_i^{\max}$, which is smaller than b_κ^*. Therefore, there is an excess task size at the κth device $b_\kappa^{\text{e}} = b_\kappa^* - b_\kappa^+$ that can not be uploaded, and must be re-allocated for processing among other offloading destinations of the ith active device. As in (16), we equally re-allocate this excess task size among the ith active device and its remaining offloading destinations whose indices are in the set $\mathcal{K}_i - \{\kappa\}$. However, it is likely that increasing the task sizes at those destinations under fixed transmit power $\mathbf{p}^*$ may increase the task uploading time enough to violate constraint (12d). Therefore, we continue re-allocating the task sizes $\mathbf{b}_i^*\ \forall i \in \mathcal{I}$ in an iterative manner until constraint (12d) is satisfied for all active devices and their offloading destinations. Once the feasible task partitioning is obtained with respect to $\mathbf{p}^*$, we can then calculate the computation resources $\mathbf{f}^*$ for the corresponding new task sizes as in (13) and (14). The overall heuristic resource initialization strategy is given in Algorithm 2.

Algorithm 2 Initial Resource Allocation

1: Calculate $\mathbf{b}^*$ and $\mathbf{p}^*$ based on (16) and (17)
2: **for** $i \in \mathcal{I}$ **do**
3: Assign a temporary set $\overline{\mathcal{K}}_i = \mathcal{K}_i$
4: Calculate $t_\kappa^{\text{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}, \forall \kappa \in \overline{\mathcal{K}}_i$
5: **if** $t_\kappa^{\text{up}^*} > \alpha t_i^{\max}, \forall \kappa \in \overline{\mathcal{K}}_i$ **then**
6: Set $t_\kappa^{\text{up}^*} = \alpha t_i^{\max}$ and update b_κ^* as $b_\kappa^+ = R_\kappa^* \alpha t_i^{\max}$
7: Calculate the excess task size $b_\kappa^{\text{e}} = b_\kappa^* - b_\kappa^+$
8: Update $\overline{\mathcal{K}}_i \leftarrow \overline{\mathcal{K}}_i - \{\kappa\}$ to partition b_κ^{e}
9: $b_i^+ = b_i^* + \frac{b_\kappa^{\text{e}}}{|\overline{\mathcal{K}}_i|+1}$
10: $b_\kappa^+ = b_\kappa^* + \frac{b_\kappa^{\text{e}}}{|\overline{\mathcal{K}}_i|+1}\ \forall \kappa \in \overline{\mathcal{K}}_i$
11: Update $\mathbf{b}_i^* \leftarrow \mathbf{b}_i^+$ then calculate $t_\kappa^{\text{up}^*}, \forall \kappa \in \overline{\mathcal{K}}_i$
12: Go to line 5
13: **end**
14: **end**
15: Calculate $\mathbf{f}^*$ by using $\mathbf{b}^*$ and $\mathbf{p}^*$ based on (13) and (14)

B. Re-allocating the Excess Resources

After running Algorithm 2, the offloaded tasks at the ES or the fog devices may not be processed if the required computation resources exceed their maximum limit, i.e., $\|\mathbf{f}_0^*\|_1 > f_0^{\max}$, where $\mathbf{f}_0^* = [f_{10}^*\ f_{20}^*\ \dots\ f_{I0}^*]^\top$, or $f_\kappa^* > f_\kappa^{\max}\ \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i - \{0\}$, respectively. Since multiple tasks from the active devices are simultaneously processed at the ES, we first focus on the violation of constraint (9c) while assuming the fog devices have infinite computation resources. In that case, the surplus task sizes at the ES must be re-allocated to the other offloading destinations. To determine what should be removed from the ES, we introduce the vector $\bar{\mathbf{r}}$ whose elements are inversely proportional to the amount of the computation resources that should be allocated to the ES for the corresponding task sizes $b_{i0}^*, \forall i \in \mathcal{I}$:

$$\bar{\mathbf{r}} = \frac{\mathbf{r}^*}{\|\mathbf{r}^*\|_1} \tag{18}$$

where $\mathbf{r}^* = \|\mathbf{f}_0^*\|_1\mathbf{1} - \mathbf{f}_0^*$. By using $\bar{\mathbf{r}}$, we can calculate the new allocated computation resources at the ES by removing the excess resources as follows:

$$\mathbf{f}_0^+ = \mathbf{f}_0^* - f^{\text{e}}\bar{\mathbf{r}} \tag{19}$$

where $f^{\text{e}} = \|\mathbf{f}_0^*\|_1 - f_0^{\max}$ contains the excess computation resources at the ES. Note that $\mathbf{f}_0^+$ in (19) not only satisfies constraint (9c), but also each element $f_{i0}^+, i \in \mathcal{I}$ of $\mathbf{f}_0^+$ is still proportional to the amount of task sizes of the active devices, which is important in terms of reducing the energy consumption as shown in Appendix B. Finally, we obtain the task sizes that should be left at the ES based on $\mathbf{f}_0^+$. Specifically, given the data rate R_{i0}^*, we calculate the new task size to be offloaded to the ES from the ith device by using (5) and (6) as follows:

$$b_{i0}^+ = \frac{R_{i0}^* f_{i0}^+ t_i^{\max}}{f_{i0}^+ + c_i R_{i0}^*},\ \forall i \in \mathcal{I} \tag{20}$$

Nevertheless, if the amount of subtracted computation resources surpasses the previously allocated resources by the ES, then some of the elements of $\mathbf{f}_0^+$ in (19) might become negative, and constraint $(9c)$ remains violated since the excess resource f^{e} could not be properly removed from $\mathbf{f}_0^*$.

In Algorithm 3, we present an iterative strategy to address the above issue and re-allocate the resources by properly removing the excess computation resource f^{e} until it becomes zero. Specifically, in line 2, we first calculate the excess computation resources to obtain $\mathbf{f}_0^+$ (19), which is the new allocated computation resources by the ES to not violate constraint (9c). If $f_{i0}^+ \geq 0, \forall i \in \mathcal{I}$, we can directly determine the corresponding task sizes that can be uploaded to the ES $b_{i0}^+,\ \forall i \in \mathcal{I}$ (20) based on $\mathbf{f}_0^+$ as given in line 15. Then, the excess task size b_i^{e}, which cannot be processed at the ES, is re-allocated among all the offloading destinations of the ith device except the ES as given by lines 16 and 17. If the new task uploading time exceeds the deadline, i.e., $\frac{b_\kappa^+}{R_\kappa^*} > t_i^{\max}$, as controlled in line 18, we run lines 5-13 in Algorithm 2 without (w/o) line 15 as it is now redundant. Hence, we obtain the final task splitting decision of the ith device by $\boldsymbol{b}_i^+ = [b_i^+\ b_{i0}^+\ b_{i1}^+\ ...\ b_{iK}^+]^\top$ to form $\mathbf{b}^+$. However, if $f_{i0}^+ < 0$ in $\mathbf{f}_0^+$ for any $i \in \mathcal{I}$, we first take its absolute value and add it to the remaining excess resource f^{e} to be removed in the next iterations as it is not properly removed from the resources that the ES initially allocated while obtaining $\mathbf{f}_0^+$ in line 2. This emphasizes that the active devices whose corresponding computation resources allocated by the ES becomes negative cannot utilize the ES for task offloading anymore. Hence, we set $f_{i0}^+ = 0$ if $f_{i0}^+ < 0, i \in \mathcal{I}$ as given in 6. Since those devices are not utilizing the ES, their initial allocated transmit power, i.e., P_{i0}^*, for the ES also becomes redundant. Therefore, we should re-allocate the transmit powers of these active devices among their other offloading destinations as given in line 8 by setting the 1st element of $\mathbf{h}_i$ to zero and using (17). In line 10, we replace $\mathbf{f}_0^*$ with $\mathbf{f}_0^+$ to be used in the next iteration in the case of $f^{\mathrm{e}} \neq 0$. Thus, Algorithm 3 recursively continues in this manner until $f^{\mathrm{e}} = 0$ and obtains $\mathbf{f}_0^+$, whose elements are the new computation resources allocated by the ES without violating constraint (9c). Finally, it outputs the new task splitting decisions $\mathbf{b}^+$, hence, we can calculate the corresponding computation resources $\mathbf{f}^+$ as in (13) and (14) by using $\mathbf{b}^+$ and $\mathbf{p}^*$.

At this point, the calculated computation resources $\mathbf{f}^+$ satisfy constraint (9c), however, constraint (9d) might be violated due to the limitation at the fog devices. Therefore, in step 25, we check this constraint for every fog device, and if it is violated, we calculate the maximum task size b_κ^m that can be processed at these fog devices as given in step 26. Then, we calculate the excess task size b_i^e in step 27 and assign it for local processing in step 28. Finally, we update the allocated task size b_κ^+, which is now feasible for processing at the fog devices.

Algorithm 3 Re-allocating the Excess Resources

1: **if** $\|\mathbf{f}_0^*\|_1 > f_0^{\max}$ **then**
2: Calculate $f^{\mathrm{e}} = \|\mathbf{f}_0^*\|_1 - f_0^{\max}$ and $\bar{\mathbf{r}}$ (18) to obtain $\mathbf{f}_0^+$ (19)
3: Reset $f^{\mathrm{e}} = 0$
4: **if** $f_{i0}^+ < 0, i \in \mathcal{I}$ **then**
5: Update $f^{\mathrm{e}} \leftarrow f^{\mathrm{e}} + |f_{i0}^+|$
6: Set $f_{i0}^+ = 0$
7: Set the ith element of $\mathbf{r}^*$ to 0
8: Set the 1st element of $\mathbf{h}_i$ to 0 and re-calculate $\mathbf{p}_i^*$ (17)
9: **end**
10: Replace the previous allocation $\mathbf{f}_0^* \leftarrow \mathbf{f}_0^+$
11: **if** $f^{\mathrm{e}} \neq 0$ **then**
12: Re-calculate $\bar{\mathbf{r}}$(18) to obtain new $\mathbf{f}_0^+$ based on $\mathbf{f}_0^*$(19)
13: Go to line 3
14: **end**
15: Based on $\mathbf{f}_0^+$, obtain $b_{i0}^+,\ \forall i \in \mathcal{I}$ (20)
16: Calculate the excess task size $b_i^{\mathrm{e}} = b_{i0}^* - b_{i0}^+, \forall i \in \mathcal{I}$
17: Re-allocate b_i^{e} among $\overline{\mathcal{K}}_i = \mathcal{K}_i - \{i0\}, \forall i \in \mathcal{I}$ for $\mathbf{b}^+$
$b_i^+ = b_i^* + \frac{b_i^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1}, \forall i \in \mathcal{I}$
$b_\kappa^+ = b_\kappa^* + \frac{b_i^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1},\ \kappa \in \overline{\mathcal{K}}_i, \forall i \in \mathcal{I}$
18: **if** $\frac{b_\kappa^+}{R_\kappa^*} > t_i^{\max},\ \kappa \in \overline{\mathcal{K}}_i, \forall i \in \mathcal{I}$ **then**
19: Update $\mathbf{b}_i^* \leftarrow \mathbf{b}_i^+$
20: Run lines 4-13 in Algorithm 2 w/o line 15
21: Update $\mathbf{b}_i^+ \leftarrow \mathbf{b}_i^*$
22: **end**
23: Calculate $\mathbf{f}^+$ by using $\mathbf{b}^+$ and $\mathbf{p}^*$ based on (13) and (14)
24: **end**
25: **if** $f_\kappa^+ > f_\kappa^{\max}, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i - \{0\}$ **then**
26: Calculate the maximum task size that can be processed $b_\kappa^{\mathrm{m}} = \frac{f_\kappa^{\max} R_\kappa^* t_i^{\max}}{R_\kappa^* c_i + f_\kappa^{\max}}$
27: Calculate the excess task size $b_i^{\mathrm{e}} = b_\kappa^+ - b_\kappa^{\mathrm{m}}$
28: Add the excess task size for local processing $b_i^+ \leftarrow b_i^+ + b_i^{\mathrm{e}}$ to update f_i^+ based on (14)
29: Update $b_\kappa^+ \leftarrow b_\kappa^{\mathrm{m}}$
30: **end**

C. Complexity Analysis

In this section, we calculate the number of operations, specifically, real multiplications and real additions, required throughout the proposed heuristic method. For simplicity, we assume that the maximum number of task offloading destinations for any ith device is $|\mathcal{K}_i| = K$. First, we consider Algorithm 2, which takes $I(K+1)$ and $I(5K+1)$ operations for (16) and (17), respectively. Note that the data rates $R_\kappa^*,\ \forall \kappa \in \mathcal{K}_i$ can be obtained in advance, which requires $4K$ operations $\forall i \in \mathcal{I}$, hence, line 4 requires only K operations. Assuming the *if condition* between the lines 5 and 13 is repeated $\forall \kappa \in \mathcal{K}_i$, we have $2K^2 + 5K$ operations. Furthermore, line 15 takes $3K + 2$ operations and repeating all these operations I times yields the time complexity of Algorithm 2 as $\mathcal{O}(IK^2 + IK)$.

Second, we consider Algorithm 3 and begin with line 2 by calculating f^{e}, $\bar{\mathbf{r}}$ (18), and $\mathbf{f}_0^+$ (19), which take $I+1$, $3I$ and $2I$ operations, respectively. Then, assuming lines 4-14 are

repeated for every active device, we have $5I^2 + 5IK + 2I$ operations. The rest of the required operations from line 15 to line 17 are $10I + 3K$ and from line 18 to line 20 are $2K^2 + 5K$. Furthermore, assuming lines 25 to 30 are repeated for all active devices and their offloading destinations, the number of required operations is equal to $9IK$. Hence, the time complexity of Algorithm 3 is $\mathcal{O}(I^2 + K^2 + IK)$.

Finally, the overall time complexity of the proposed heuristic method is $\mathcal{O}(IK^2 + I^2 + K^2)$.

V. Simulation Results

In this section, we examine the performance of both the convex-programming method and the heuristic method through Monte-Carlo simulations in a variety of scenarios. As a benchmark we use a lower bound on the total energy consumption derived in Appendix B. For the network layout, we uniformly distribute the active devices (AD)s within a 500×500 m^2 area and the fog devices (FD)s are placed on a disk with a radius of 15m centered at an active device. We consider independent Rayleigh fading for all communication links and we use the path loss model $\text{PL}_{\text{cell}} = 128.1 + 37.6\log_{10}(d)$, where d is the distance in km [37], for the cellular wireless links between ADs and the BS, while for the D2D links between ADs and FDs, we use $\text{PL}_{\text{D2D}} = 148 + 40\log_{10}(d)$. At each simulation run, the task size $d_i,\ \forall i \in \mathcal{I}$, is chosen from a uniform distribution $\mathcal{U}(2\times10^4,\ 4\times10^5)$. To better illustrate the performance of the proposed methods, we consider the challenging situation where conditions (9c) and (9d) in the original problem are violated by limiting the maximum computation capability of the ES $f_0^{\max}$ to

$$f_0^{\max} = \eta\mathbb{E}\left[\sum_{i\in\mathcal{I}} f_i^{\text{opt}}\right] \tag{21}$$

and the maximum capability of the FDs to

$$f_\kappa^{\max} = \eta\mathbb{E}\left[f_i^{\text{opt}}\right],\ \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i \tag{22}$$

where f_i^{opt} is obtained from Appendix B and $\eta \in [0, 1]$ is a scaling term. In other words, we ensure that problem $\mathscr{P}_3$ and Algorithm 3 are always employed in the simulations for both proposed methods. Specifically, we choose $\eta = 0.8$ to reduce the computation capabilities by 20% of the average total required computation resources in the ideal scenario given in Appendix B. The rest of the system parameters are given in Table I unless specified otherwise.

TABLE I. System parameters

Parameter Description	Symbol	Value
Network size	-	500×500 m^2
Max. radius of a D2D link	-	25 m
Number of active devices	I	$\{2, 3, \ldots, 12\}$
Number of fog devices	K	$\{0, 1, \ldots, 5\}$
Task size	d_i	$[2\times10^4,\ 4\times10^5]$ bits
CPU cycles to process 1-bit data	c_i	1500 cycles/bit
Effective capacitance constant	μ	10^{-24} Ws^3
Scaling term for $f_0^{\max}$	η	$[0.8, 1]$
Limiting term for task uploading time	α	0.85
Max. computation capability at the ES	$f_0^{\max}$	$[0.2,\ 1.5]$ GHz
Max. transmit power	$P^{\max}$	$[0,\ 200]$ mW
Task processing deadline	$t_i^{\max}$	$[0.4,\ 1]$ s
Noise level	N_0	-114 dBm
Channel bandwidth	W	10 MHz

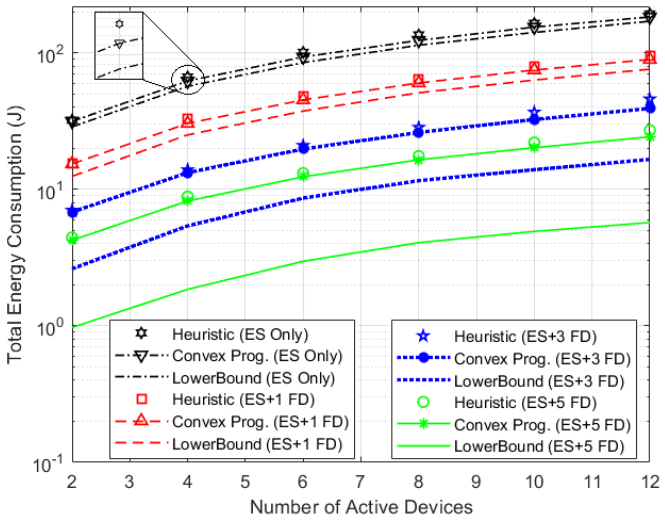


Fig. 2. Total energy consumption versus different number of ADs while $t_i^{\max} = 1\ \forall i \in \mathcal{I}$ and $P^{\max} = 200$ mW.

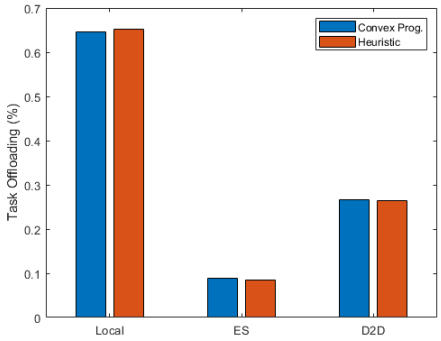
In Fig. 2, we show the total energy consumption as a function of the number of ADs $I \in \{2, 4, \ldots, 12\}$ and the number of FDs $K \in \{0, 1, 3, 5\}$. As the number of ADs changes, we set the computation capability of the ES and the FDs by using (21) and (22). We compare both proposed methods to the lower bound on the total energy consumption obtained in Appendix B. As we can see, by incorporating D2D communications in the task offloading process, the total energy consumption is reduced compared to utilizing only the ES, and decreases as the number of fog devices increases.

We note that the performance of both proposed methods starts deviating from the lower bound as the number of FDs increases. This happens for two reasons. First, because the computation capability of the ES is finite and due to constraint (9c), task partitioning cannot be equally distributed as in Appendix B. Second, task uploading time is not instantaneous as opposed to the ideal scenario in Appendix B. Hence, more computation resources must be allocated to complete the tasks within the deadline in order to meet constraints (9f) and (9g). Consequently, the total energy consumption deviates from the lower bound. In what follows, we gradually relax the constraints in our simulations to validate these explanations and show that the proposed methods achieve near-optimal performance.

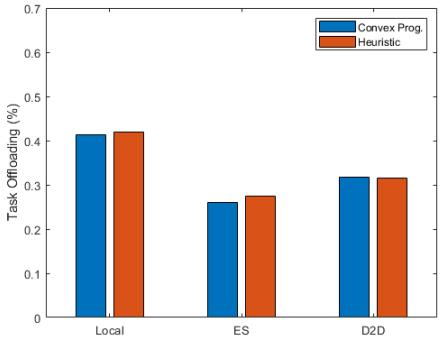
To validate the first reason, in Fig. 3 we show the effect of maximum computation resources on the total energy consumption by plotting the utilization of the offloading destinations in percentage for a given task under various values of $f_0^{\max}$ and $f_\kappa^{\max}$. The task offloading ratio is defined as follows:

$$\text{Task Offloading (\%)} = \underbrace{\frac{1}{I}\sum_i \frac{b_i^+}{d_i}}_{\text{Local}} + \underbrace{\frac{1}{I}\sum_i \frac{b_{i0}^+}{d_i}}_{\text{ES}} + \underbrace{\frac{1}{I}\sum_i \frac{\frac{1}{K}\sum_{\kappa\in K_i-\{0\}} b_\kappa^+}{d_i}}_{\text{D2D}} \tag{23}$$

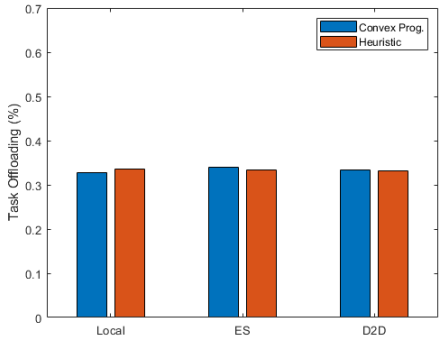
where $b_\kappa^+, \forall \kappa \in \mathcal{K}_i \; i \in \mathcal{I}$ is obtained from Section III and Section IV for the convex programming method and the heuristic method, respectively. We set $f_0^{\max} \in \{.2, .4, .8\}$GHz and $\eta = \{0.8, 0.95, 1\}$, while we choose $I = 5$, $|\mathcal{K}_i| \in \{0, 1\}$ and $t_i^{\max} = 1 \; \forall i \in \mathcal{I}$. If both the convex-programming method and the heuristic method allocate the tasks as equal as possible while constraints (9c) and (9d) are relaxed, we can show that the proposed methods can approach the optimal solution given as the lower bound.



(a) When $f_0^{\max} = 0.2$ GHz and $\eta = 0.8$ used in (22)



(b) When $f_0^{\max} = 0.4$ GHz and $\eta = 0.95$ used in (22)



(c) When $f_0^{\max} = 0.8$ GHz and $\eta = 1$ used in (22)

Fig. 3. Utilization of the offloading destinations in percentage under different computation capability of the ES.

In Fig. 3a, when $f_0^{\max}$ is 0.2 GHz and $\eta = 0.8$, the utilization of the ES and the FDs are limited, hence, most of the tasks are processed at the local device. In this case, the performance deviations from the lower bound for the convex programming and the heuristic method are 51% and 52%, respectively. When the limitation of computation resources at the ES and the FDs are gradually relaxed by choosing $f_0^{\max} = 0.4$GHz and $\eta = 0.95$ as in Fig. 3b, the participation of the ES and FDs for task offloading increases, which reduces the performance gap to 17% for the convex-programming and 20% for the heuristic method. Finally, when there is no resource limitation at the ES and the FDs (relatively to task sizes and the number of devices in the network), both convex-programming and the heuristic methods achieve almost equal task size allocation among the active devices and their offloading destinations, which yields near-optimal solution as the performance deviations are only 0.009% and 0.019%, respectively.

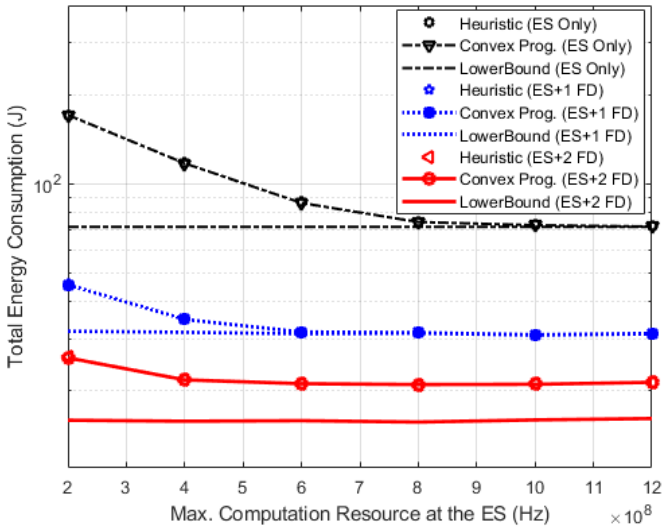


Fig. 4. Total energy consumption versus maximum computation capability of the ES, i.e., $f_0^{\max}$.

To validate the second reason, we study the effect of the limitation of data rates and transmit powers while changing the number of FDs. We assume that the number of ADs is 6 and we set $\eta = 0.8$ while increasing $f_0^{\max}$ as in Fig. 4. When the computation capability of the ES is $f_0^{\max} = 0.2$ GHz, the performance gap for both proposed methods in the case of utilizing only the ES (black line) is significantly higher compared to incorporating a single FD (blue line) as the limitation of $f_0^{\max}$ is being compensated with the additional computation resources at the FD. However, for values of $f_0^{\max}$ higher than 0.6 GHz, the effect of $f_0^{\max}$ almost disappears and both proposed methods approach to a near-optimal solution for these two scenarios. Nonetheless, with the addition of one more FD (red line), a constant offset appears between the performance of the proposed methods and the lower bound, which is a direct indication that the devices require more computation resources to compensate for the task uploading time as they try to meet constraints (9f) and (9g). Therefore, we numerically demonstrated that both proposed methods achieve a near-optimal solution and the performance gaps occur mostly due to the imposed constraints rather then their sub-optimal nature.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3190085

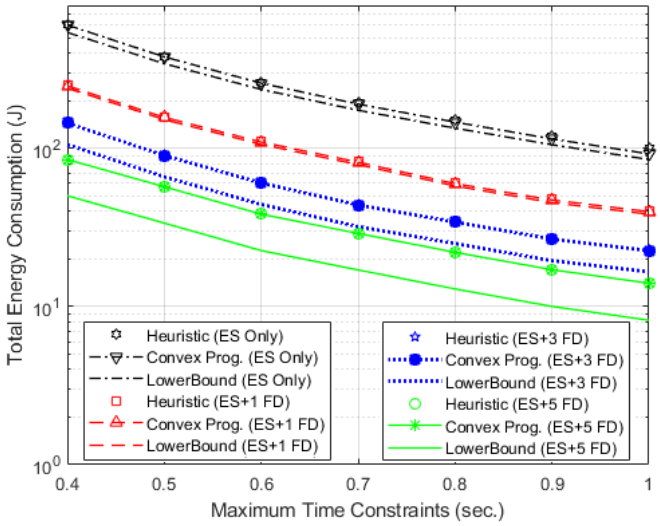


Fig. 5. Total energy consumption versus different maximum time constraints, i.e., $t_i^{\max} \in [0.4,\ 1]\ \forall i \in \mathcal{I}$, while $I = 6$ and $P^{\max} = 200$.

In Fig. 5, we plot the total energy consumption as a function of the task processing deadline. Similar to the result in Fig. 4, as $t_i^{\max}\ \forall i \in \mathcal{I}$ decreases, more computation resources must be allocated by the devices to process the offloaded tasks within the given deadline, and consequently, the total energy consumption increases. However, it can be also verified that both the convex-programming method and the heuristic method in the case of utilizing only the ES or a single FD achieve near-optimal performance regardless of the task processing deadline, which again demonstrates that the performance gaps are due to the imposed constraints as including more FDs increases them.

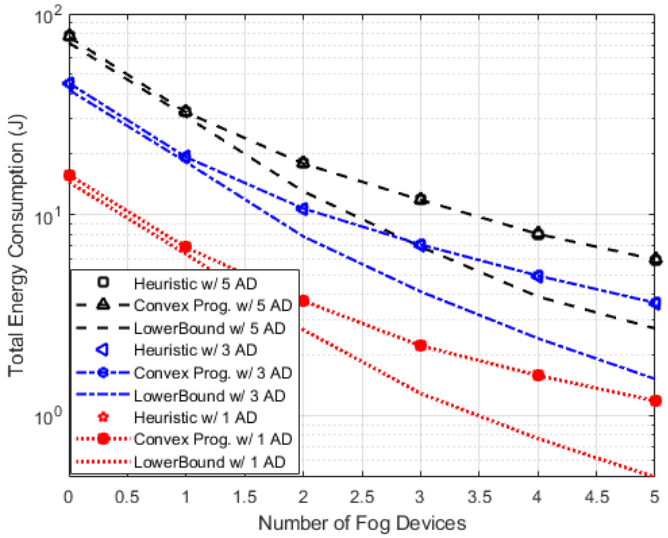


Fig. 6. Total energy consumption with respect to different number of ADs and IDs while $t_i^{\max} = 1$ and $P^{\max} = 200$.

In Fig. 6, we study the total energy consumption with respect to the change in the number of fog devices, where $K \in \{0, 1, \ldots, 5\}$ and $I \in \{1, 3, 5\}$. Similar to Fig. 2, the computation capability of the ES is determined based on (21) as the number of devices in the network change in each simulation run. It is shown that by increasing the number of FDs in the task offloading process, the total energy consumption can be significantly decreased. Specifically, it takes ten times less energy to compute a similar task with the help of 5 FDs via D2D links instead of utilizing only the ES. Nevertheless, the total energy consumption is higher in the case of 1 AD compared to the case of 5 ADs and 5 FDs. This shows that taking a computation intensive task off a single device and effectively partitioning it across all the available computation resources can significantly reduce the overall energy consumption even though more devices are involved in the task offloading process.

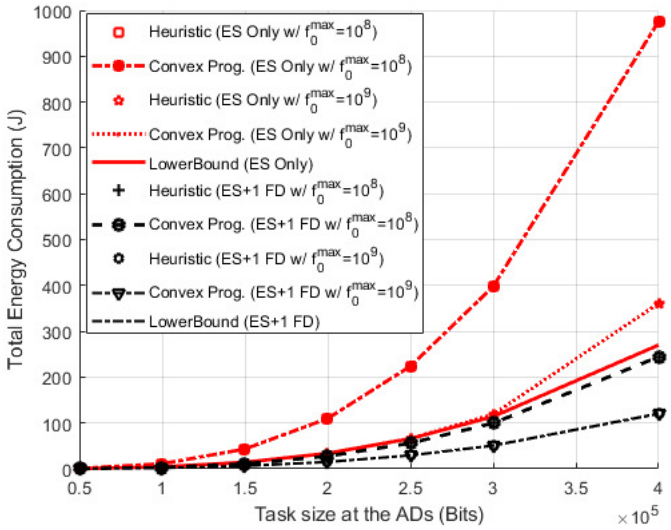


Fig. 7. Total energy consumption versus task size under different number of offloading devices and computation capability of the ES, i.e., $|\mathcal{K}_i| \in \{0, 1\}$ and $f_0^{\max} \in \{10^8, 10^9\}$, respectively.

In Fig. 7, we study the total energy consumption for the different task sizes. The number of ADs is set to 5 while there is only a single FD. In addition, we set the available computation resource of the ES to pre-defined values as $f_0^{\max} \in \{10^8, 10^9\}$. It is shown that when the task size is 400kbits, there is a drastic increase in the total energy consumption if D2D communication is not used for task offloading, especially when $f_0^{\max} = 10^8$Hz. On the contrary, by employing only a single FD in the task offloading process, the total energy consumption decreases by almost 5 times. More importantly, the change in the total energy consumption with $f_0^{\max} = 10^8$Hz and $f_0^{\max} = 10^9$Hz is relatively small when compared to utilizing only the ES. Hence, we can conclude that by incorporating D2D communication and exploiting nearby computation resources, we can alleviate the dependence on the ES, which is crucial when the traffic density is high and the available resources are scarce in the network.

In Fig. 8, we compared the performance of our proposed methods to a benchmark study [33] in terms of energy efficiency. The main reason for choosing [33] is because it uses the same objective function, i.e., minimization of total energy consumption. In addition, similar to our model, [33] assumes a centralized cloud server utilized through a BS and multiple D2D devices for task offloading. However, in

contrast to our work, where we assume a centralized ES with a limited computation capability, no constraint is put on the computational capability of the cloud server. Also, power control and local task processing are not considered during task offloading, which drastically increases the energy efficiency regardless of the number of FDs and the maximum time constraints chosen for the comparison. Therefore, the proposed methods outperforms [33] while achieving near-optimal solution.

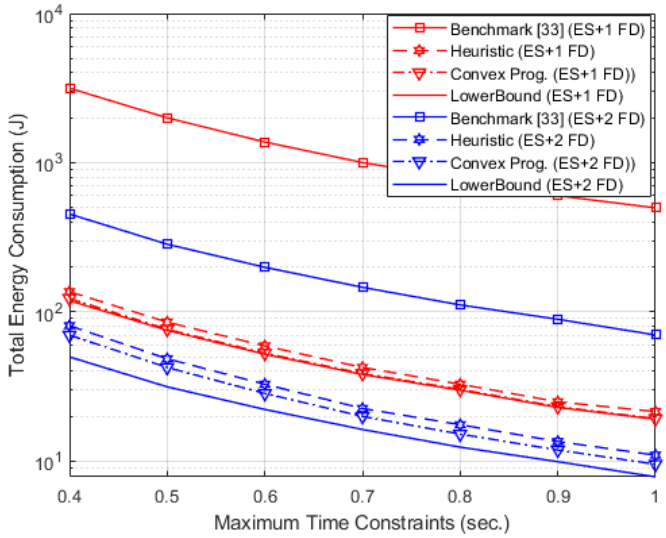


Fig. 8. Total energy consumption versus different maximum time constraints, i.e., $t_i^{\max} \in [0.4,\ 1]\ \forall i \in \mathcal{I}$, while $I = 6$ and $P^{\max} = 200$.

Finally, we compare the average run-time of both proposed methods implemented in MATLAB and executed on an Intel i7-3770 computer with 16GB RAM. For the first comparison, we assume only one AD and three FDs while changing the task processing deadline from 0.4 to 1 second as presented in Table II. Even though the proposed convex programming method slightly outperforms the heuristic one in terms of energy efficiency as demonstrated earlier, its average run-time is significantly higher. For the second comparison as shown in Table III, we include four more ADs to the network while keeping the number of FDs the same. For $t^{\max} = 0.4$s, the average run time of our heuristic method is 7.6x10^3 times less than the convex programming method.

TABLE II. Average run-time comparison when $I = 1$ and $J = 3$

Simulation setup	Convex Prog.	Heuristic
$t^{\max} = 0.4$ s	0.385 s	161.1 μs
$t^{\max} = 0.6$ s	0.371 s	160.4 μs
$t^{\max} = 0.8$ s	0.366 s	160.2 μs
$t^{\max} = 1$s	0.349 s	159.8 μs

TABLE III. Average run-time comparison when $I = 5$ and $J = 3$

Simulation setup	Convex Prog.	Heuristic
$t^{\max} = 0.4$ s	3.818 s	0.501 ms
$t^{\max} = 0.6$ s	3.708 s	0.449 ms
$t^{\max} = 0.8$ s	3.429 s	0.448 ms
$t^{\max} = 1$s	2.847 s	0.448 ms

VI. Conclusion

In this paper, we addressed a resource allocation problem in a multi-device D2D-aided fog computing scenario, wherein a central ES and proximate fog devices are utilized for task offloading. Since the formulated problem is intractable, we first proposed a sub-optimal convex-programming based method, in which the computation resources, task sizes and the transmit powers are allocated to reduce the total energy consumption in the network. Then, based on this first method, we developed a heuristic task offloading method, which does not require the computation of gradients and Hessian matrices during the solution process. We showed in detail the computational complexity of this method in terms of key system parameters, including the number of mobile devices and task offloading destinations. Finally, we developed a lower bound on the total energy consumption as a performance benchmark for both the convex-programming and the heuristic methods. Computer simulations demonstrated that the proposed methods significantly reduce the total energy consumption compared to processing tasks only locally while attaining near-optimal solution in comparison to the derived lower bound.

Appendix A
Convexity of Problem $\mathscr{P}_2$

We begin with proving the each function that forms the total computation energy given in (11) is convex. Since the summation of convex functions is a convex function, without loss of generality, we can consider an active device and a single offloading device $\kappa \in \mathcal{K}_i = \{i0\}$ to prove the convexity of (11) as follows:

$$\psi_i(\mathbf{p}_i, \mathbf{b}_i) = \underbrace{\frac{\mu(b_i c_i)^3}{\left(t_i^{\max}\right)^2}}_{\psi_{\text{loc}}} + \underbrace{\frac{\mu(b_\kappa c_i)^3}{\left(t_i^{\max} - \frac{b_\kappa}{R_\kappa}\right)^2}}_{\psi_{\text{off}}},\ \forall i \in \mathcal{I}$$

We denote the Hessian of ψ_{loc} and ψ_{off} by $\Delta^2\psi_{\text{loc}}$ and $\Delta^2\psi_{\text{off}}$, respectively. Obtaining $\Delta^2\psi_{\text{loc}}$ is straightforward since it only depends on b_i. Hence, after calculating its eigenvalues as 0 and $\frac{6b_i\mu c_i^3}{(t_i^{\max})^2}$, we can determine that $\Delta^2\psi_{\text{loc}}$ is positive semi-definite and ψ_{loc} is convex since $b_i \geq 0,\ \forall i \in \mathcal{I}$. For the Hessian of ψ_{off}, we have:

$$\Delta^2\psi_{\text{off}} = \begin{bmatrix} \frac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa^2} & \frac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa \partial P_\kappa} \\ \frac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa \partial b_\kappa} & \frac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa^2} \end{bmatrix}$$

where we can show that its trace and determinant are positive to prove the convexity of ψ_{off}. Specifically, the first element of the main diagonal is equal to:

$$\frac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa^2} = \frac{6\mu b_\kappa^3 c_i^3}{R_\kappa^2\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2} + \frac{12\mu b_\kappa^2 c_i^3}{R_\kappa\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2} + \frac{6\mu b_\kappa c_i^3}{\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2}$$

while the second entry is:

$$\frac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa^2} = \frac{m_i b_\kappa^4}{u\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^3} + \frac{2m_i b_\kappa^4}{u_\kappa \ln(v_\kappa)\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^3} + \frac{3b_\kappa^5 m_i \ln(2)}{W u_\kappa \ln^2(v)\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^4}$$

where $m_i = 2\mu c_i^3 G_\kappa^2 \ln(2)$, $v_\kappa = (1 + \frac{P_\kappa G_\kappa}{N_0})$ and $u_\kappa = W {N_0}^2 v_\kappa^2 \ln^2(v_\kappa)$ are positive variables. Since the task uploading time is always smaller than the task processing deadline due to constraint (15e), i.e., $t_\kappa^{\text{up}} \leq \alpha t_i^{\max}$, the trace of $\Delta^2\psi_{\text{off}}$ is positive.

Furthermore, the determinant of $\Delta^2\psi_{\text{off}}$ is calculated as:

$$\det(\Delta^2\psi_{\text{off}}) = \frac{4\mu^2 c_i^6 b_\kappa^5 G_\kappa^2 \ln(2) s_i}{N_0^2 W^3 \ln^5(v_\kappa) v_\kappa^2 \big(t_i^{\max} - t_\kappa^{\text{up}}\big)^7}$$

where

$$s_i = b_\kappa^2 \ln^2(2) + W^2 t_i^{\max} \ln^2(v_\kappa)\big(6t_i^{\max} - 7t_\kappa^{\text{up}} + 3t_i^{\max}\ln(v_\kappa)\big)$$

We can show that $\det(\Delta^2\psi_{\text{off}})$ is positive if s_i and $(t_i^{\max} - t_\kappa^{\text{up}})$ are positive. Both can be ensured simultaneously if $t_i^{\max} > 7/6 t_\kappa^{\text{up}}$. Hence, we must select $\alpha \in (0, \frac{6}{7})$ in constraint (15e) to keep the determinant positive for any values of b_κ and P_κ, $\forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I}$. Therefore, ψ_{off} is a convex function over the convex set based on the constraints in problem $\mathscr{P}_2$. Finally, we can conclude that $\psi_i(\mathbf{p}_i, \mathbf{b}_i)$, $\forall i \in \mathcal{I}$ is convex as it is a summation of convex functions.

Appendix B
Optimal Task Offloading for the Ideal Case

We calculate the lower bound on the total energy consumption (8) by evaluating the optimal task offloading strategy in an ideal scenario, where the task uploading time is instantaneous and the offloading destinations have infinite amount of computation resources. As demonstrated in *Lemma 1* in Section III-A, the task size d_i, regardless of how it is split, must be computed at exactly $t_i^{\max}$ to minimize energy consumption. Hence, we can calculate the total resource required for computing d_i as follows:

$$f_i^{\text{tot}} = \frac{d_i c_i}{t_i^{\max}} \tag{24}$$

At this point, the main objective becomes finding the optimal task splitting strategy, i.e, $d_i = b_i + \sum_{\kappa \in \mathcal{K}_i} b_\kappa$, such that the allocated computation resources, i.e., f_i and f_κ $\forall \kappa \in \mathcal{K}_i$, minimize the total energy consumption in (8).

Suppose that there is no constraint on the computation capability of device $k \in \mathcal{K}_i$ and the task uploading speed is very high. Hence, we can assume $P_\kappa = 0$ and $t_\kappa^{\text{up}} = 0$, which yields $t_i^{\text{co}} = t_\kappa^{\text{co}} = t_i^{\max}$, $\forall i \in \mathcal{I}$ $\forall \kappa \in \mathcal{K}_i$. Then, based on (2) and (6), we have $b_i = \frac{f_i t_i^{\max}}{c_i}$ and $b_\kappa = \frac{f_\kappa t_i^{\max}}{c_i}$, $\forall \kappa \in \mathcal{K}_i$, equivalently, the total required resource is $f_i^{\text{tot}} = f_i + \sum_{\kappa \in \mathcal{K}_i} f_\kappa$. Therefore, by using (3) and (7), the total energy consumption of the ith device is equal to:

$$E_i = \mu f_i^3 t_i^{\max} + \sum_{\kappa \in \mathcal{K}_i} \mu f_\kappa^3 t_i^{\max} \tag{25}$$

To obtain the lower bound for (25), we define the following problem:

$$\min_{\mathbf{f}} \quad \sum_{i \in \mathcal{I}} E_i \tag{26}$$

$$\text{s.t.} \quad f_i + \sum_{\kappa \in \mathcal{K}_i} f_\kappa = f_i^{\text{tot}}, \ \forall i \in \mathcal{I} \tag{27}$$

Since the problem is convex, we can find the optimal value by using Lagrange multiplier method, where the Lagrangian function for the ith device with a Lagrange multiplier λ_i is defined as follows:

$$\mathcal{L}_i(\mathbf{f}_i, \lambda_i) = E_i - \lambda_i (f_i + \sum_{\kappa \in \mathcal{K}_i} f_\kappa - f_i^{\text{tot}}) \tag{28}$$

Then, by taking the gradient of (28) and solving it as $\nabla \mathcal{L}_i(\mathbf{f}_i, \lambda_i) = 0$, we have:

$$3\mu \begin{bmatrix} f_i^2 \\ f_{i0}^2 \\ f_{i1}^2 \\ \vdots \\ f_{iK}^2 \end{bmatrix} t_i^{\max} - \lambda_i \mathbf{1} = 0 \tag{29}$$

where $\mathbf{1}$ is the column vector of all-ones. Hence, we obtain $f_i = f_\kappa = \sqrt{\frac{\lambda_i}{3\mu t_i^{\max}}}$ $\forall \kappa \in \mathcal{K}_i$, and by using (27), we have $\lambda_i = 3\mu t_i^{\max} \big(\frac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}\big)^2$. Consequently, the optimal computation resources are calculated as $f_i = f_\kappa = \frac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}$, $\forall \kappa \in \mathcal{K}_i$. Finally, by considering (24), the optimal task splitting based on (2) and (6) becomes $b_i = b_\kappa = \frac{d_i}{|\mathcal{K}_i|+1}$.

This shows that the minimum total energy consumption is achieved when the tasks are split equally. Accordingly, the allocated computation resource at each device should be identical, where we can denote the optimum resource as $f_i^{\text{opt}} = \frac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}$. Therefore, the lower bound of the total energy consumption for the given task size d_i can be calculated as follows:

$$E_i^* = \mu\big(|\mathcal{K}_i| + 1\big)\big(f_i^{\text{opt}}\big)^3 t_i^{\max} \leq E_i \tag{30}$$

References

[1] Cisco Annual Internet Report, 2018–2023 [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

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

[3] X. Chen, Z. Liu, Y. Chen and Z. Li, "Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks," *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[4] Y. Siriwardhana, P. Porambage, M. Liyanage and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects," *IEEE Commun. Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[5] M. Huang, W. Liu, T. Wang, A. Liu and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Int. of Things Jour.*, vol. 7, no. 7, pp. 5792–5805, 2020.

[6] J. Yan, S. Bi, Y. J. Zhang and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 1, pp. 235–250, 2020.

[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 & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Jour. on Selec. Areas in Commun.*, vol. 34, no. 12, pp. 3590–3605, 2016.

[9] Y. Fan, L. Zhai and H. Wang, "Cost-efficient dependent task offloading for multiusers," *IEEE Access*, vol. 7, pp. 115843–115856, 2019.

[10] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE Int. Symp. on Quality of Service*, pp. 1–10. June 2017.

[11] C. Wang, F. R. Yu, C. Liang, Q. Chen and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. on Vehic. Tech.*, vol. 66, no. 8, pp. 7432–7445, 2017.

[12] X. Chen, Z. Liu, Y. Chen and Z. Li, "Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks," *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[13] C. Zhao, Y. Cai, A. Liu, M. Zhao and L. Hanzo, "Mobile edge computing meets mmWave communications: Joint beamforming and resource allocation for system delay minimization," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 4, pp. 2382–2396, 2020.

[14] L. Liu, Z. Chang, X. Guo, and T. Ristaniemi, "Multi-objective optimization for computation offloading in mobile-edge computing," in *Proc. IEEE Symp. Comput. Commun.*, pp. 832–837, Jul. 2017.

[15] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne and L. Hanzo, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," *IEEE Trans. on Vehic. Tech.*, vol. 69, no. 2, pp. 2246–2262, 2020.

[16] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," in *IEEE/ACM Trans. on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[17] N. Li, J. Martinez-Ortega and V. H. Diaz, "Distributed power control for interference-aware multi-user mobile edge computing: A game theory approach," *IEEE Access*, vol. 6, pp. 36105–36114, 2018.

[18] B. Wu, J. Zeng, L. Ge, Y. Tang and X. Su, "A game-theoretical approach for energy-efficient resource allocation in MEC network," in *Proc. IEEE Int. Conf. on Commun.*, pp. 1–6, May 2019.

[19] H. Hong, "From cloud computing to fog computing: unleash the power of edge and end devices," in *Proc. IEEE Int. Conf. on Cloud Comp. Tech. and Sci.*, pp. 331–334, Dec. 2017.

[20] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[21] J. Wen, C. Ren and A. K. Sangaiah, "Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 96–102, 2018.

[22] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein and F. H. P. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[23] Y. He, J. Ren, G. Yu and Y. Cai, "D2D Communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. on Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, 2019.

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

[25] L. Pu, X. Chen, J. Xu and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Jour. on Selec. Areas in Commun.*, vol. 34, no. 12, pp. 3887–3901, 2016.

[26] B. Gao, Z. Zhou, F. Liu, F. Xu and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. on Mobile Comp.*, doi: 10.1109/TMC.2021.3064847.

[27] C. Yi, S. Huang and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Trans. on Mobile Comp.*, vol. 20, no. 3, pp. 1076–1091, 2021.

[28] L. Li, L. Gu, J. Hong and S. Jiang, "Joint computation offloading and wireless resource allocation in mobile edge computing," in *IEEE Proc. Int. Conf. on Comp. and Commun.*, pp. 705–711, Dec. 2018.

[29] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[30] Z. Ning, P. Dong, X. Kong and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Int. of Things Jour.*, vol. 6, no. 3, pp. 4804–4814, June 2019.

[31] X. Meng, W. Wang and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.

[32] S. Yu, R. Langar and X. Wang, "A D2D-multicast based computation offloading framework for interactive applications," in *IEEE Proc. Glob. Commun. Conf.*, pp. 1-6, Dec. 2016.

[33] W. Liu, Y. Teng, M. Liu and M. Song, "Joint offloading and computation resource allocation in D2D assisted hybrid framework," in *IEEE Proc. Annual Int. Symp. on Pers., Indoor and Mobile Radio Commun.*, pp. 1–6, Sept. 2019.

[34] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. on Net.*, vol. 27, no. 1, pp. 85–97, 2019.

[35] P. Mach, Z. Becvar, and T. Vanek, "In-band device-to-device communication in OFDMA cellular networks: A survey and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1885–1922, 4th Quart., 2015.

[36] S. Sharma, N. Gupta, and V. A. Bohara, "OFDMA-based device-to device communication frameworks: Testbed deployment and measurement results," *IEEE Access*, vol. 6, pp. 12019–12030, 2018.

[37] Y. Dai, M. Sheng, K. Zhao, L. Liu, J. Liu and J. Li, "Interference-aware resource allocation for D2D underlaid cellular network using SCMA: A hypergraph approach," in *Proc. IEEE Wireless Commun. and Network Conf.*, pp. 1–6, Apr. 2016.